Neural Machine Translation for Low-Resource Scenarios

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

M.Sc. Software Systems Engineering
Yunsu Kim
aus Seoul, Südkorea

Berichter: Universitätsprofessor Dr.-Ing. Hermann Ney
Universitätsprofessor Dr. Alfons Juan Císcar

Tag der mündlichen Prüfung: 7. Februar 2022

There are things that never change.

Acknowledgments

First of all, I would like to thank my supervisor, Prof. Dr.-Ing. Hermann Ney, who offered me this great opportunity to study interesting topics and conduct state-of-the-art research. He gave me considerate and significant advice during the Ph.D. course, which was crucial in finishing the thesis work; I will always remember that advice for the rest of my career.

Also, I would like to express my gratitude to Prof. Dr. Alfons Juan, who has kindly agreed to be the second referee of this thesis. Whenever we met at a conference, in Valencia, and at Aachen, his kind words and supportive feedback on the research were helpful and inspiring.

Next, I am truly grateful to the Bachelor/Master students I worked with: Jiahui Geng, Hendrik Rosendahl, Miguel Graça, Arne Nix, Duc Thanh Tran, Petre Petrov, Benedikt Hilmes. They were all diligent and enthusiastic about academic research and team work. Without their great contributions, this thesis would never have been completed. I also learned a lot from them and gained valuable experience as a supervisor.

It cannot be said too often how important my colleagues at Lehrstuhl Informatik 6, RWTH Aachen University are in this thesis work. I deeply thank all colleagues who accompanied me on this journey: Andreas Guta, Jörn Wübker, Weiyue Wang, Yingbo Gao, Jan Rosendahl, Christian Herold, Julian Schamper, Malte Nuhn, Jan-Thorsten Peter, Tamer Alkhouli, Stephan Peitz, Markus Freitag, Nick Rossenbach, Parnia Bahar, David Thulke, Markus Kitza, Harald Hanselmann, Kazuki Irie, Gabriel Bretschner, Oscar Koller, Julian Dierkes, Christoph Lüscher, Wilfried Michel, Tina Raissi, Peter Vieting, Albert Zeyer, Wei Zhou, Eugen Beck, Patrick Dötsch, Pavel Golik, Mahdi Hamdani, Tobias Menne, Zoltán Tüske, Ilya Kulikov, Evgeniia Torkachuk, Stefan Koltermann, Thomas Dackweiler, Ralf Schlüter, and Volker Steinbiss.

During my time as Ph.D. candidate, I had an excellent opportunity to work jointly with eBay Research in Aachen, which motivated many topics in this thesis work. I am grateful for enormous support from Shahram Khadivi, Pavel Petrushkov, Leonard Dahlmann, Michael Kozielski, and José G. C. de Souza. Also, I should not forget the cooperation with AppTek, especially with Evgeny Matusov and Patrick Wilken, where I learned a lot in practical system building and joint research.

I would also like to give special thanks to Stephanie Jansen for her sincere caring about the logistics of my Ph.D. defense and the thesis publication.

Needless to say, nearly ten years of studying abroad could never have been successful without the continuous belief and support from my parents and friends in my home country, South Korea. I have been and will be grateful to them forever.

Finally, I dedicate this thesis to my wife, Minduck Kim, who endured numerous difficulties and obstacles, consistently supported me directly and indirectly, showed a great deal of love and sacrifice throughout the time of my Ph.D. course. Thank you so much.

Abstract

Machine translation has been tackled for decades mainly by statistical learning on bilingual text data. In the most recent paradigm with neural network modeling, building a machine translation system requires more data than ever to make the best use of the state-of-the-art modeling capacity and yield a reasonable performance. Unfortunately, however, there is not a sufficient amount of bilingual corpora for many language pairs and domains. To expand the coverage of neural machine translation, this thesis investigates effective methods to improve the performance in such low-resource scenarios.

Firstly, we study the usage of monolingual corpora for neural machine translation. To begin with, we optimize the log-linear integration of a language model into translation decoding. Next, we review various synthetic data generation strategies and compare their empirical performance at scale. In addition, we investigate pre-training and multi-task training of a translation model with language modeling and the Cloze task modeling objectives. We compare all these methods empirically to provide best practice for semi-supervised learning to compensate for the performance in a low-resource case.

Secondly, we examine the cross-lingual transfer from a high-resource setting to a low-resource setting. This study covers two pragmatic scenarios: transfer between the language pairs whose target side is common and transfer from multiple language pairs based on a pivot language, e.g. for a non-English language pair with English as the pivot. For both scenarios, we develop a series of sequential transfer techniques to maximize the effectiveness of the transfer. The techniques are thoroughly compared to semi-supervised baselines, multilingual models and cascaded architectures.

Lastly, we investigate unsupervised learning for neural machine translation, where only monolingual corpora are used to train a translation model. We cover the methods from classical decipherment to sequence-to-sequence training, giving a historical overview of unsupervised translation. For decipherment, we extend its primitive framework to large vocabulary translation by reducing lexicon sizes in training and employing neural network lexicons. Furthermore, we integrate a cross-lingual word embedding lexicon model and apply a neural denoising autoencoder as a postprocess, leading to a novel cascaded combination. Finally, we analyze the most sophisticated method with a sequence-to-sequence model, including extensive experimental results on numerous data settings to find out under which conditions unsupervised learning is useful in practice.

Kurzfassung

Die maschinelle Übersetzung wird seit Jahrzehnten hauptsächlich durch statistisches Lernen zweisprachiger Textdaten angegangen. In dem jüngsten Paradigma mit neuronalen Netzen erfordert der Aufbau eines maschinellen Übersetzungssystems mehr Daten als je zuvor, um die hochmoderne Modellierungskapazität optimal zu nutzen und eine angemessene Leistung zu erzielen. Leider gibt es für viele Sprachpaare und Domänen nicht genügend zweisprachige Korpora. Um die Abdeckung der neuronalen maschinellen Übersetzung zu erweitern, werden in dieser Arbeit effektive Methoden zur Verbesserung der Leistung in solchen ressourcenarmen Szenarien untersucht.

Zunächst untersuchen wir die Verwendung einsprachiger Korpora für die neuronale maschinelle Übersetzung. Wir optimieren die logarithmische lineare Integration eines Sprachmodells in die Übersetzungsdecodierung. Als Nächstes überprüfen wir verschiedene Strategien zur Erzeugung synthetischer Daten und vergleichen ihre empirische Leistung im Maßstab. Darüber hinaus untersuchen wir das Vorlernen und das Multitask-Lernen eines Übersetzungsmodells mit Sprachmodellierung und den Cloze-Task-Modellierungszielen. Wir vergleichen alle diese Methoden empirisch, um die beste Vorgehensweise für halbüberwachtes Lernen zur Kompensation der Leistung in einem Fall mit geringen Ressourcen bereitzustellen.

Zweitens untersuchen wir den mehrsprachigen Transfer von einer Einstellung mit hohen Ressourcen zu einer Einstellung mit niedrigen Ressourcen. Diese Studie deckt zwei pragmatische Szenarien ab: Übertragung zwischen den Sprachpaaren, deren Zielseite gemeinsam ist, und Übertragung von mehreren Sprachpaaren basierend auf einer Pivot-Sprache, z.B. für ein nicht englisches Sprachpaar mit Englisch als Pivot. Für beide Szenarien entwickeln wir eine Reihe von sequentiellen Übertragungstechniken, um die Effektivität der Übertragung zu maximieren. Die Techniken werden gründlich mit halbüberwachten Baselines, mehrsprachigen Modellen und kaskadierten Architekturen verglichen.

Zuletzt untersuchen wir unbeaufsichtigtes Lernen für die neuronale maschinelle Übersetzung, bei der nur einsprachige Korpora zum Trainieren eines Übersetzungsmodells verwendet werden. Wir behandeln die Methoden von der klassischen Entschlüsselung bis zum Sequenz-zu-Sequenz-Training und geben einen historischen Überblick über die unbeaufsichtigte Übersetzung. Zur Entschlüsselung erweitern wir sein primitives Framework auf die Übersetzung großer Vokabeln, indem wir die Lexikongrößen im Training reduzieren und Lexika für neuronale Netze verwenden. Darüber hinaus integrieren wir ein mehrsprachiges Lexikonmodell zur Worteinbettung und wenden einen Autoencoder mit neuronaler Entrauschung als Nachbearbeitung an, was zu einer neuartigen kaskadierten Kombination führt. Dann analysieren wir die ausgefeilteste Methode zum Erlernen eines Sequenz-zu-Sequenz-Modells, einschließlich umfangreicher experimenteller Ergebnisse zu zahlreichen Dateneinstellungen, um herauszufinden, unter welchen Bedingungen unbeaufsichtigtes Lernen in der Praxis nützlich ist.

Contents

1. Introduction

In this thesis, we investigate various methodologies for low-resource scenarios in machine translation, especially when working with neural translation models (Section 3.3). Firstly, in this chapter, we review the concept of machine translation (Section 1.1) and define what a low-resource scenario in machine translation is (Section 1.2). We also preview each part of the thesis briefly in Section 1.3.

1.1 Machine Translation

Machine translation (MT) is the task of converting given text from one language to another language using automatic algorithms. In contrast to human translations, once it is built, an MT system can be used repeatedly for many purposes without additional labor costs. This can help a lot to facilitate communication between people from different countries, e.g. in sharing knowledge or in traveling around [Way 13].

However, the quality of MT outputs is still behind human translations except for some popular language pairs and domains [Toral & Castilho$^+$ 18, Läubli & Sennrich$^+$ 18, Graham & Haddow$^+$ 19, Barrault & Bojar$^+$ 19]. Translation is known to be inherently a hard problem due to the linguistic dissimilarity between languages [Knight 99, Zens & Ney 03, Koehn 10, Hirschmann & Nam$^+$ 16]. Figure 1.1 shows an example of translating an English sentence to Korean. There are numerous factors to consider in the translation process: 1) "change" is at the end of the English sentence but appears in the middle in Korean (long-range reordering). 2) "There are" is better translated as a single phrase instead of each word at a time (phrase translation). 3) The order of "There are" and "things" is switched when translating them into Korean (local reordering). 4) "never" is responsible for not just one word but multiple lexical tokens in Korean, i.e. "절대" and "않" (one-to-many alignment). 5) "세상엔" is not directly generated from any words in the English sentence but rather added for fluency (target insertion). Moreover, we should conjugate and rearrange the tokens correctly within a context, solving lexical ambiguities and preserving syntactic integrity at the same time; the task is far beyond simple substitutions of words.

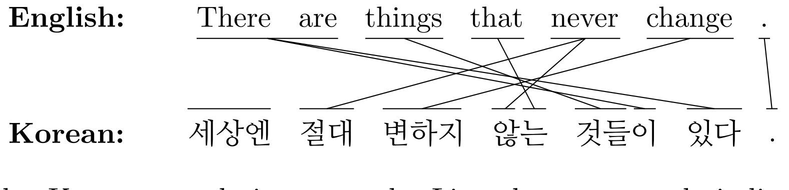


Figure 1.1: English→Korean translation example. Lines between words indicate that they are semantically related to each other.

One possible solution to this complex task is applying case-by-case translation rules in the correct order [Vauquois 68, Hutchins 86, Nirenburg 89]. Each rule is written by a bilingual human expert and defines e.g. lexical translation or reordering for a specific case. Such rule-based (knowledge-based) systems were dominant in MT before the 1990s; however, they are not scalable since it is too costly to write all rules for all language pairs considered [Koehn 10, Costa-Jussa & Farrús$^+$ 12].

Statistical approaches to MT have replaced rule-based MT thanks to their efficiency in system building and improved performance. Mathematical models are trained with bilingual corpora, i.e. sets of sentence pairs in two different languages which are translations of each other. Once we have such data, building an MT system is fully computational and does not need any human expertise. Statistical MT can be categorized according to the lexical unit of the main model component: word-based [Brown & Cocke$^+$ 90, Brown & Della Pietra$^+$ 93], phrase-based [Zens & Och$^+$ 02, Koehn & Och$^+$ 03], and sequence-to-sequence [Bahdanau & Cho$^+$ 15, Vaswani & Shazeer$^+$ 17]. The sequence-to-sequence family is also referred to as neural machine translation, because its model is based solely on artificial neural networks.

Neural machine translation (Section 3.3) is the state-of-the-art statistical approach at this moment, exploiting richer context with better sharing of representations over similar inputs [Koehn 20]. It has been used in almost all submissions to shared tasks for MT organized by the Conference on Machine Translation (WMT) in recent years [Bojar & Chatterjee$^+$ 17, Bojar & Federmann$^+$ 18, Barrault & Bojar$^+$ 19], which is a clear indicator of its popularity and effectiveness. Unless otherwise noted, the methods discussed in this thesis are all developed for the neural machine translation framework.

1.2 Low-Resource Scenario in Machine Translation

Statistical MT is data-driven; the performance of an MT system relies on the size and quality of the training data, i.e. bilingual sentence pairs. Such bilingual corpora are very scarce for most language pairs and typically limited to a couple of domains, e.g. news. For popular language pairs like German-English or Chinese-English, one can train a decent MT system with millions of bilingual sentence pairs for a data-rich domain, which is sometimes claimed to reach even human parity [Hassan & Aue$^+$ 18, Barrault & Bojar$^+$ 19]. However, when we are given only a handful of bilingual data, e.g. less than 100k sentence pairs, the training procedure cannot collect sufficient statistics for translating unseen input sentences, if no special care is taken in compensating for the data scarcity. Accordingly, the MT performance tends to be far from a reasonable quality, especially for neural translation models [Koehn & Knowles 17].

In this thesis, we designate such a difficult data condition as a *low-resource* scenario in MT. More specifically, in building an MT system for a translation task with specific test sets, we define a low-resource scenario as follows:

1. There are *less than 300k* bilingual sentence pairs of fine quality available that can be used for training translation model components. A fine sentence pair should consist of semantically and syntactically equivalent sentences in source and target languages, where liberal translations are also allowed. Noisy pairs with wrong translations or missing parts are not counted in defining the low-resource condition.
2. The performance of the MT system on the desired test sets is *below 20% in* Bleu score (Section 3.6.3) when it is built by the standard supervised training (Section 3.3.2) with only the given bilingual training corpora.

How small the training data must be and how bad the supervised baseline system should perform for a task to be classified as a low-resource scenario are subjective depending on the expectations on the MT quality and the progress of the MT research. However, from the WMT shared task definitions, we can get a general idea of the low-resource conditions for the worldwide research

Table 1.1: Low-resource scenarios in WMT news translation tasks. Test performance is evaluated on the newstest set of the corresponding year, using a base Transformer model [Vaswani & Shazeer$^+$ 17] trained with supervised learning (Section 3.3.2).

Year	Language pair	Bilingual training data		Test performance
		#sentences	#words (English)	Bleu [%]
2017	Turkish-English	207,678	4,428,278	19.2
2019	Kazakh-English	222,424	1,717,393	10.3
2019	Gujarati-English	155,798	1,352,186	9.9

community, which reflect the state-of-the-art status of the MT techniques. Table 1.1 shows several translation tasks from WMT which are described as challenging due to their small bilingual training data size. According to the statistics of these tasks, we set the thresholds for a low-resource translation task for this thesis as above. If we do not have any bilingual data for training, the task is called a *zero-resource* scenario.

1.3 About This Thesis

This thesis tackles the poor performance of MT in low-resource scenarios. An intuitive solution is to collect more bilingual corpora, which is done by either letting humans translate monolingual documents [Koehn 05, Tiedemann 12, Ziemski & Junczys-Dowmunt$^+$ 16] or crawling a massive amount of comparable corpora on the web [Smith & Saint-Amand$^+$ 13, Schwenk & Chaudhary$^+$ 19, Esplà-Gomis & Forcada$^+$ 19]. The former costs a lot of human labor and the latter requires a tremendous effort to filter noisy examples [Koehn & Khayrallah$^+$ 18, Koehn & Guzmán$^+$ 19]. Crawling and filtering of bilingual corpora are also important topics of MT research, but their technical background often deviates significantly from the computational modeling for MT [Moore 02, Resnik & Smith 03, Varga & Kornai$^+$ 05, Esplá-Gomis & Forcada 09, Taghipour & Khadivi$^+$ 11, Ștefănescu & Ion$^+$ 12, Kaufmann 12, Etchegoyhen & Azpeitia 16, Xu & Koehn 17]. Moreover, the data collection is highly dependent on the language pair and domain.

In this thesis, we focus on using already available data sources which are not necessarily bilingual of the desired language pair, e.g. monolingual data. These additional sources do not involve extra procedures like translation or bilingual sentence matching. Also, the methods using these sources are not orthogonal to the technical details of the MT modeling and training. In fact, they give a deeper understanding of how the current MT model architecture reacts to other types of training data.

The following chapters assume three conditions on the training data. Chapter 4 is about *semi-supervised* learning, where monolingual data of either input or output side are added to given bilingual data. Chapter 5 deals with *cross-lingual* learning, where bilingual corpora of other language pairs are used along with the desired language pair. Chapter 6 examines an extreme case of training MT models only with monolingual data, which is called *unsupervised* learning. In each of these chapters, we study state-of-the-art techniques to make use of the corresponding additional data source and verify their effectiveness with comprehensive experiments. Note that the methods studied in the thesis were originally developed for low-resource scenarios but they are applicable also to data-rich settings, though their effect on translation performance might be smaller.

Before we begin with the core methods, we first define the goals of this thesis scientifically in Section 2. The goals are checked again in Chapter 7 to assess to what extent they are accomplished at the end of the thesis work. For background knowledge of MT, we refer the reader to Chapter 3.

2. Scientific Goals

The goal of this thesis is to answer the following research questions in building an MT system for a low-resource scenario using neural networks:

1. What is the best way to utilize **monolingual corpora** along with given bilingual corpora? (Chapter 4)
    - What are the possible ways to generate synthetic bilingual sentence pairs from monolingual corpora?
    - How can we train a translation model with monolingual training objectives? How much does it improve the translation performance?
    - How can we effectively integrate a pre-trained language model into the MT decoding?
2. What is the best way to utilize **bilingual corpora of other language pairs**? (Chapter 5)
    - What are the possible ways to share the same model parameters over different language pairs?
    - When training a single model for multiple language pairs, how can we alleviate language dissimilarity and maximize knowledge sharing?
    - How can we alleviate the discrepancy of model components that are trained independently with different training corpora?
    - How can we avoid pivoting and build a strong single model for a non-English language pair?
3. What is the best way to train neural translation models **only with monolingual corpora**, given no bilingual corpora at all? (Chapter 6)
    - How can we extend the traditional decipherment framework to integrate neural network lexicon models?
    - How can we build an MT system efficiently without iterative training in an unsupervised setup?
    - Under which conditions does unsupervised learning produce comparable performance to supervised or semi-supervised learning in MT?

3. Preliminaries

Prior to the core research of this thesis, this chapter explains the basics of statistical/neural MT, which serve as baselines in our experiments. Terminologies and notations defined in this chapter are used consistently throughout the rest of this thesis. From the next chapters, we assume the readers know the materials of this chapter.

3.1 Terminologies and Notations

As described in Section 1.1, a pair of sentences in two different languages is considered in a translation task. An input sentence to an MT system is in the *source* language and its output sentence is in the *target* language. Such a bilingual sentence pair is represented as sequences of words:

$$f_1^J = f_1, ..., f_j, ..., f_J \qquad \text{source sentence} \tag{3.1}$$

$$e_1^I = e_1, ..., e_i, ..., e_I \qquad \text{target sentence} \tag{3.2}$$

where f_j denotes a source word and e_i a target word. J and I are the lengths of the source/target sentence, respectively. We use $\mathcal{C}_{f,e}$ to indicate a corpus of sentence pairs (f_1^J, e_1^I). For a set of monolingual sentences, e.g. $\{f_1^J\}$, we use $\mathcal{C}_f$ with a single subscript.

A bilingual corpus is normally given without the information of which target word is the translation of which source word. Word *alignment* means how each target word is aligned to the related source words, and vice versa. Our notation for source-to-target alignments is:

$$a_1^J = a_1, ..., a_j, ..., a_J \tag{3.3}$$

where a_j denotes the target word positions aligned to f_j:

$$a_j \subseteq \{1, ..., I\} \quad \text{or} \quad a_j = \{0\} \tag{3.4}$$

Position 0 stands for an empty target word, thus $\{0\}$ indicates that a source word is not aligned to any target word. Word alignments can be automatically learned using statistical models [Och & Ney 03]. Note that, depending on the preprocessing of bilingual corpora, these notations might be at the subword or character level. Figure 1.1 gives an example of subword-level alignments.

A simpler type of bilingual resource is a bilingual dictionary. A dictionary $\mathcal{D}_{f,e}$ contains bilingual word pairs (f, e): a source word f and a target word e that can be the translation of each other. In Chapter 6, we work a lot with bilingual dictionaries when it is too difficult to build a translation model directly at the sentence level with unsupervised learning.

The methods covered in this thesis are all based on probability theory. Throughout this thesis, Pr stands for a true distribution which ideally accounts for statistics over the whole universe, and p denotes a model distribution which is parametrized to approximate the true distribution. We use θ to indicate the parameters to be trained, where p_θ or $p(\cdot;\theta)$ means that the model p is parametrized by θ.

3.2 Statistical Machine Translation

The goal of statistical MT can be mathematically defined as finding the best translation hypothesis $\hat{e}_1^{\hat{I}}$ in the target language which maximizes the source→target translation probability $Pr(e_1^I|f_1^J)$:

$$f_1^J \mapsto \hat{e}_1^{\hat{I}}(f_1^J) = \operatorname*{argmax}_{I,\, e_1^I} \left\{ Pr(e_1^I|f_1^J) \right\} \tag{3.5}$$

Based on this decision rule, there are three main problems of statistical MT:

1. **Modeling** is about formulating $Pr(e_1^I|f_1^J)$ and designing a mathematical model which approximates the unknown true distribution. A model consists of a set of parameters and their relations, which should be reasonable to explain the translation process.
2. **Training** a model means that we optimize the model parameters to a certain criterion, e.g. in order that given training data are most probable under the model.
3. **Decoding** is a task of searching for the maximum probable output sequence of a model, i.e. performing the argmax operation over I and e_1^I. Since the search space of e_1^I is usually huge, decoding involves pruning techniques to avoid iterating over unlikely outputs for efficiency.

Each problem has its own goal and methodologies, while some solutions to a problem are related to another problem. For example, one can model $Pr(e_1^I|f_1^J)$ with a neural network with softmax output layers (Section 3.3.1), which is better to be trained with the cross-entropy criterion (Section 3.3.2). However, it is important not to lump these problems together; one should pinpoint the exact weakness of an MT system and concentrate on that problem. For instance, even if the training method is greatly improved, it is of no worth if the model is found to be too weak. When presenting any idea in this thesis, we strictly indicate which problem it is trying to solve.

In the following, we explore two classical approaches to modeling $Pr(e_1^I|f_1^J)$.

3.2.1 Noisy-Channel Modeling

The first successful model for statistical MT [Brown & Cocke$^+$ 90] reformulates the source-to-target translation probability $Pr(e_1^I|f_1^J)$ with Bayes' rule before finding a suitable model function:

$$\operatorname*{argmax}_{I,\, e_1^I} \left\{ Pr(e_1^I|f_1^J) \right\} = \operatorname*{argmax}_{I,\, e_1^I} \left\{ \frac{Pr(e_1^I, f_1^J)}{Pr(f_1^J)} \right\} \tag{3.6}$$

$$= \operatorname*{argmax}_{I,\, e_1^I} \left\{ \frac{Pr(e_1^I) \cdot Pr(f_1^J|e_1^I)}{Pr(f_1^J)} \right\} \tag{3.7}$$

$$= \operatorname*{argmax}_{I,\, e_1^I} \left\{ Pr(e_1^I) \cdot Pr(f_1^J|e_1^I) \right\} \tag{3.8}$$

$$= \operatorname*{argmax}_{I,\, e_1^I} \left\{ p(e_1^I) \cdot p(f_1^J|e_1^I) \right\} \tag{3.9}$$

where $Pr(f_1^J)$ is dropped since it is not dependent on I and e_1^I. In the end, we model the target language probability with $p(e_1^I)$ and the target-to-source translation probability with $p(f_1^J|e_1^I)$ separately. This is called *noisy-channel* modeling [Shannon 48], assuming that the target sentence e_1^I is distorted into the source sentence f_1^J by a noisy communication channel $p(f_1^J|e_1^I)$. The task is now to decode f_1^J back to e_1^I with the help of a prior distribution of the original sentence $p(e_1^I)$.

In the early days of statistical MT, translation modeling was at an elementary level and bilingual resources were extremely scarce; thus direct modeling of $Pr(e_1^I|f_1^J)$ was not reliable. In the noisy-channel model, the target language model (LM) $p(e_1^I)$ is able to compensate for the weakness of the translation model $p(f_1^J|e_1^I)$. It is possible to build a powerful LM independently with vast monolingual corpora, which are easier to obtain than bilingual data. A strong LM allows $p(f_1^J|e_1^I)$ to be less expressive, e.g. decomposed into word level with omitting some conditional dependencies.

Noisy-channel modeling has been popular in various natural language processing (NLP) tasks due to its effectiveness with an assist from an LM, e.g. speech recognition [Bahl & Jelinek$^+$ 83], spelling correction [Kernighan & Church$^+$ 90, Church & Gale 91, Mays & Damerau$^+$ 91], and question answering [Echihabi & Marcu 03]. In MT, word-based IBM models [Brown & Cocke$^+$ 90, Brown & Della Pietra$^+$ 93] are based on the noisy-channel formulation, in which the translation model is extended by including word alignments and length variables. In Section 6.2, we adopt the noisy-channel modeling for unsupervised MT.

Noisy-channel modeling is mathematically well-defined as a product of probability distributions, which corresponds to a chain of generative steps [Koehn 10]. This assumes correct normalization of each model component as well as the final product of components, which theoretically guarantees the optimum by maximizing $Pr(e_1^I) \cdot Pr(f_1^J|e_1^I)$. However, in practice, we work with model distributions $p(e_1^I)$ and $p(f_1^J|e_1^I)$, whose product may not necessarily yield the best performance [Och & Ney 02]. Also, it is not straightforward to extend the model with other variables that might be of interest in the translation process, e.g. linguistic tags. The IBM models show that it is already very complex to include a few latent variables in the noisy-channel model.

3.2.2 Log-Linear Modeling

In phrase-based MT [Zens & Och$^+$ 02, Koehn & Och$^+$ 03], *log-linear* modeling is usually used to include many different features in a model. It directly approximates $Pr(e_1^I|f_1^J)$ as follows:

$$\begin{aligned}
\operatorname*{argmax}_{I,e_1^I} \left\{ Pr(e_1^I|f_1^J) \right\} &= \operatorname*{argmax}_{I,e_1^I} \left\{ p(e_1^I|f_1^J) \right\} &\quad (3.10)\\
&= \operatorname*{argmax}_{I,e_1^I} \left\{ \frac{\exp\left(\sum_{m=1}^{M} \lambda_m h_m(f_1^J, e_1^I)\right)}{\sum_{I',e'^{I'}_1} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(f_1^J, e'^{I'}_1)\right)} \right\} &\quad (3.11)\\
&= \operatorname*{argmax}_{I,e_1^I} \left\{ \exp\left(\sum_{m=1}^{M} \lambda_m h_m(f_1^J, e_1^I) \right) \right\} &\quad (3.12)\\
&= \operatorname*{argmax}_{I,e_1^I} \left\{ \sum_{m=1}^{M} \lambda_m h_m(f_1^J, e_1^I) \right\} &\quad (3.13)
\end{aligned}$$

where h_m is a *feature* and λ_m is its weight. In this framework, we project each bilingual sentence pair as a vector of features, which are computed individually and combined later as if they are independent of one another. Note that h_m is not necessarily a probability distribution; it can be any type of scoring function, e.g. an unnormalized count or a sparse binary feature, which may or may not be a parametrized model. Typical features of a phrase-based MT system are phrase translation models, word translation models, reordering models, and a target LM. Each weight denotes how responsible the feature is for the translation process.

The combined scores are renormalized over all possible hypotheses (Equation 3.11), but the denominator can be ignored in the maximization (Equation 3.12). Applying logarithms makes the model a linear combination of features (Equation 3.13), thus called log-linear modeling. One can easily integrate a new feature into the model by adding it to the total score, ignoring the normalization constraint. The model can be even enriched with thousands of features [Chiang & Knight$^+$ 09, Hopkins & May 11, Green & Wang$^+$ 13]. Due to their flexibility, log-linear models were adopted also in part-of-speech tagging [Ratnaparkhi 96], named entity recognition [Borthwick & Sterling$^+$ 98], or language modeling [Khudanpur & Wu 00].

However, each feature of a log-linear model is learned separately with a different algorithm, possibly on a distinct dataset, which makes the entire training complex and difficult to maintain. Also, features are designed solely by human experts, which is laborious and depends highly on the properties of each language pair. Another tricky part of using a log-linear model is the estimation of the scaling weights λ_m. For a large number of features, the weights cannot be manually adjusted and should be algorithmically trained to optimize the performance [Dairoch & Ratcliff 72, Och 03, Crammer & Singer 03, Crammer & Kulesza$^+$ 09, Hopkins & May 11]. Such algorithms are trained with a separate bilingual corpus which is disjunct from the one used for training the features. Note that training of the weights is not deterministic and suffers from instability; normally, multiple runs of training are conducted and the average and variance of their performance metrics (Section 3.6.3) are reported [Cer & Manning$^+$ 10, Clark & Dyer$^+$ 11].

For an overview of phrase-based MT and its features, we refer the reader to [Kim 15]. Training of the log-linear scaling weights is explained thoroughly in [Freitag 10]. We utilize log-linear modeling in Section 6.3 to combine an LM with a lexicon learned without supervision.

3.3 Neural Machine Translation

Noisy-channel (Section 3.2.1) or log-linear (Section 3.2.2) modeling assigns various linguistic aspects to different model components and combines them to build the final model. Each model component operates at the word or phrase level, and the sentential context in the output is handled only by a target LM. Meanwhile, the global context of the source sentence cannot be used effectively in traditional word-/phrase-based MT.

Artificial neural networks have emerged as a solution to the complex model building process and limited context information. Unlike count tables or heuristic features, neural networks convert discrete inputs from text into continuous representations, which has the following advantages:

- They encode arbitrarily diverse information in a high-dimensional vector space. Each feature in phrase-based MT has a specific, interpretable role in the translation process. On the other hand, a neural network learns any useful knowledge required to predict its output from the input, invisibly blended in a vector.
- They handle arbitrarily long context within the model capacity. A count-based LM must increase its table size to cover higher orders of n-grams (Section 3.4.1). However, a neural network may use the same set of parameters for processing every position of a sentence, which is efficient and effective in terms of knowledge sharing.
- They alleviate the sparsity problem with their continuous-valued nature. Count-based models usually employ smoothing techniques that manually redistribute probability mass to unseen events [Kneser & Ney 95, Chen & Goodman 99]. Neural networks have no concept of zero counts and generalize to unseen examples within a vector space, where the representations of similar words are close to each other [Bengio & Ducharme$^+$ 03, Mikolov & Chen$^+$ 13].

For a general introduction to artificial neural networks, we recommend [Goodfellow & Bengio$^+$ 16].

Early attempts at applying neural networks to MT were made by including them as additional components in the model combination, or by using them to re-rank the first-pass hypotheses from an MT system. LMs [Schwenk & Déchelotte$^+$ 06, Le & Allauzen$^+$ 10, Schwenk & Rousseau$^+$ 12] or translation models [Schwenk 12, Son & Allauzen$^+$ 12, Auli & Galley$^+$ 13, Kalchbrenner & Blunsom 13, Devlin & Zbib$^+$ 14, Sundermeyer & Alkhouli$^+$ 14] were implemented with neural networks, which improved the word-/phrase-based MT in this way. Note that, from this time onwards, translation models have often been conditioned by both source and target words, i.e. jointly modeling monolingual fluency and bilingual adequacy. In [Alkhouli 20], one can find the details of integrating the neural network model components into a phrase-based log-linear framework.

The ultimate goal of neural networks for MT is to build a single, comprehensive model which incorporates all necessary linguistic information in the translation process, e.g. lexical context, word alignment, semantics, reordering rules, etc. Avoiding the model combination framework, the state-of-the-art formulation of neural machine translation (NMT) is:

$$\underset{I,\,e_1^I}{\operatorname{argmax}} \left\{ Pr(e_0^I|f_1^J) \right\} = \underset{I,\,e_1^I}{\operatorname{argmax}} \left\{ \prod_{i=1}^{I} Pr(e_i|e_0^{i-1}, f_1^J) \right\} \tag{3.14}$$

$$= \underset{I,\,e_1^I}{\operatorname{argmax}} \left\{ \prod_{i=1}^{I} p(e_i|e_0^{i-1}, f_1^J) \right\} \tag{3.15}$$

which is factorized over target positions i. The model $p(e_i|e_1^{i-1}, f_1^J)$ predicts a target word at position i given the source sentence and the entire target history. This has no simplifying assumptions in modeling, i.e. the model distribution depends on the same conditions as the true distribution. Also, there are no additional latent variables in the model which can introduce intermediate errors. All target positions share the same model parameters for $p(e_i|e_1^{i-1}, f_1^J)$, i.e. effectively only one model component. This approach is said to be *end-to-end*; there are no mid-processes of training multiple model components and scaling weights.

The next subsections describe how to design the model $p(e_i|e_0^{i-1}, f_1^J)$ with neural networks (Section 3.3.1), how to train the network parameters (Section 3.3.2), and how to efficiently perform the argmax operation of Equation 3.15 (Section 3.3.3).

3.3.1 Encoder-Decoder Model

The standard model architecture for NMT consists of two modules: encoder and decoder. The *encoder* learns a sequence of representations $\mathbf{h}_1^J$ from source tokens f_1^J, where $\mathbf{h}_j \in \mathbb{R}^{D_h}$ is a fixed-size vector of D_h dimensions which encodes the semantic and/or syntactic relation of f_j to other source words $\{f_{j'} \,|\, j' \neq j\}$. The *decoder* takes this representation sequence as its input and computes the probability of hypothesizing target output words, modeling the relations of a target word to all source words. It predicts a target word e_i for each position i, considering previously hypothesized target tokens e_1^{i-1}:

$$\mathbf{h}_1^J = \mathrm{E}(f_1^J) \tag{3.16}$$

$$p(e_i|e_0^{i-1}, f_1^J) = \mathrm{D}(\mathbf{h}_1^J, e_1^{i-1}) \tag{3.17}$$

where this distribution is implemented as a vector of probability masses, which has the size of the target vocabulary $\mathcal{V}_e$. E and D denote the entire encoder or decoder, respectively. The two modules are connected to each other to form a final sequence-to-sequence model.

In the following, we present the technical details of each module individually. Afterwards, we explain additional architectural concepts which are common to both encoder and decoder. Our description is based on the Transformer [Vaswani & Shazeer$^+$ 17], which is the state-of-the-art NMT model at the time of writing this thesis.

Encoder

An encoder is composed of three main parts: word embedding, positional encoding, and self-attention layers.

Word Embedding. The first step of an encoder is to convert source words of a discrete form to continuous-valued vectors, also known as word embeddings. In order to facilitate this conversion in matrix-vector calculations, an embedding layer does the following in order:

1. Words are mapped onto natural numbers in a finite set of source vocabulary $\mathcal{V}_f = \{1, ..., |\mathcal{V}_f|\}$.
2. Each source word $f_j \in \mathcal{V}_f$ is represented by one-hot encoding: a vector of $|\mathcal{V}_f|$ dimensions whose f_j-th dimension is set to one but all others to zero.
3. A corresponding column in the embedding matrix $\mathbf{W}_f \in \mathbb{R}^{D_h \times |\mathcal{V}_f|}$ is selected, which stores an embedding vector for each source word.

The procedure can be summarized as follows:

$$f_j \in \mathcal{V}_f \;\longmapsto\; \delta_{d=f_j} \in \mathbb{R}^{|\mathcal{V}_f|} \;\longmapsto\; \mathrm{b}_f(f_j) \in \mathbb{R}^{D_h} \tag{3.18}$$

where b_f denotes all three steps of a source word embedding layer.

The continuous-valued space of word embedding may encode much more information on words, e.g. conceptual comparisons, than a discrete entity in the vocabulary [Mikolov & Chen$^+$ 13]. It also allows the further parts of the network to process the inputs consistently in matrix calculations. However, the embedding layer does not consider the context around a word dynamically; an embedding vector of a specific word is the same regardless of its position or neighboring words.

Positional Encoding. Along with source words, an encoder is also given their absolute positions (j), i.e. at which index each word is located in a sentence. This is crucial information about the relationships between words that characterize the syntax of the source sentence. Positional encoding (pos) represents j with a vector of D_h dimensions, which is of the same size as the word embedding. The most widely used formulation of this vector is:

$$\mathrm{pos}(j)_d = \begin{cases} \sin\left(\frac{j}{10000^{d/D_h}}\right) & d \text{ is even} \\ \cos\left(\frac{j}{10000^{(d-1)/D_h}}\right) & d \text{ is odd} \end{cases} \tag{3.19}$$

for each dimension $d = 1, ..., D_h$. Each dimension corresponds to a sinusoidal wave over j, where the frequency and offset of the wave is different for each dimension. This formulation is fixed and not trainable. Positional encoding can also be trained together with other parameters of the network [Gehring & Auli$^+$ 17], but it has been empirically found to perform similarly to the fixed formula [Vaswani & Shazeer$^+$ 17]. The positional encoding is added to the word embedding before it is processed by the upper layers:

$$\mathbf{f}_j = \mathrm{b}_f(f_j) + \mathrm{pos}(j) \tag{3.20}$$

Self-Attention. Word embeddings represent each word in a source sentence but not their relationships to each other. Self-attention layers of an encoder learn contextual dependencies of a source word on other source words. Let $\mathbf{h}_j^{(l)} \in \mathbb{R}^{D_h}$ be the result of l-th self-attention layer on j-th position and $\mathbf{H}^{(l)} \in \mathbb{R}^{D_h \times J}$ a matrix of $\mathbf{h}_j^{(l)}$'s for all source positions. Starting from $\mathbf{h}_j^{(0)} = \mathbf{f}_j$, self-attention is basically a weighted sum of the representations from the previous layer:

$$\mathbf{h}_j^{(l)} = \sum_{j'=1}^{J} \alpha_{jj'}^{(l)} \cdot \mathbf{h}_{j'}^{(l-1)} \tag{3.21}$$

where the weights are normalized probabilities over the source positions:

$$\alpha_{jj'}^{(l)} = \mathrm{softmax}(\mathbf{h}_j^{(l-1)\top} \cdot \mathbf{H}^{(l-1)})_{j'} \tag{3.22}$$

$$= \frac{\exp(\mathbf{h}_j^{(l-1)\top} \cdot \mathbf{h}_{j'}^{(l-1)})}{\sum_{j''=1}^{J} \exp(\mathbf{h}_j^{(l-1)\top} \cdot \mathbf{h}_{j''}^{(l-1)})} \tag{3.23}$$

Here, the similarity between two representation vectors is defined by the dot product. The similarity function can also be modeled with trainable weights [Bahdanau & Cho$^+$ 15, Luong & Pham$^+$ 15].

$\mathbf{h}_j^{(l-1)}$ and $\mathbf{h}_{j'}^{(l-1)}$ in Equation 3.23 are called *query* and *key*, respectively. Query is the j-th representation whose relationship to other positions (j') is asked. Key means a unique property of each position j', which is used to compute the queried relation (weight). Having quantified the relations ($\alpha_{jj'}$), we retrieve the actual representation vector of j', called *value* ($\mathbf{h}_{j'}^{(l-1)}$), whose weighted sum is calculated (Equation 3.21). It resembles key-value retrievals from a dictionary, where each key is unique and there is a corresponding value that contains the actual information for the key. In a typical NMT model, a key for j' is the representation vector itself ($\mathbf{h}_{j'}^{(l-1)}$) and equal to its value, which is simple to implement and informative enough for computing the relation to query. Note that query, key, and value have the same dimensionality D_h.

The dot product operations in a self-attention layer do not depend on one another, thus can be parallelized over j and j'. The parallel computation boils down to simple matrix multiplications:

$$H^{(l)} = \mathrm{softmax}(\mathbf{Q}_f^{(l-1)\top}\mathbf{K}_f^{(l-1)}) \cdot \mathbf{V}_f^{(l-1)} \tag{3.24}$$

where $\mathbf{Q}_f$, $\mathbf{K}_f$, and $\mathbf{V}_f$ are query, key, and value matrices with $\mathbf{Q}_f = \mathbf{K}_f = \mathbf{V}_f = \mathbf{H}^{(l-1)}$. This process is repeated for L_f layers, where the final output of an encoder for j is denoted by $\mathbf{h}_j = \mathbf{h}_j^{(L_f)}$. As an encoder gets deeper with more layers, it is capable of learning more complex relations among source words, e.g. contextual semantics and syntax [Voita & Sennrich$^+$ 19].

The sequential relations of $\mathbf{f}_j$'s can also be modeled with recurrent [Bahdanau & Cho$^+$ 15] or convolutional [Gehring & Auli$^+$ 17] layers. However, these have higher complexity in computing long-range dependencies and the recurrent unit cannot be parallelized over the positions [Vaswani & Shazeer$^+$ 17]. In the end, they usually perform worse than self-attention in many standard evaluation tasks for MT [Vaswani & Shazeer$^+$ 17, Hieber & Domhan$^+$ 17, Tang & Müller$^+$ 18].

Decoder

A decoder generates a target word for each target position i from left to right, i.e. one word at a time from $i = 1$ to $i = I$. It keeps track of the generated hypothesis up to the previous position (e_1^{i-1}) and relates the generation with source representations $\mathbf{h}_1^J$ from the encoder. This relation is represented as an internal state $\mathbf{s}_i^{(l)}$, which is computed for each decoder layer l.

At each position i, a decoder first obtains vector representations of the already generated target sequence $\mathbf{e}_1^{i-1}$, analogously to source words:

$$\mathbf{e}_i = \mathrm{b}_e(e_i) + \mathrm{pos}(i) \tag{3.25}$$

which are set as the bottommost states, i.e. $\mathbf{s}_i^{(0)} = \mathbf{e}_{i-1}$. Note that their positions are shifted to the right by one. We introduce an artificial target position $i = 0$ and assign a sentence start symbol to it, i.e. $e_0 =$ `<s>`, which offers the initial context when generating the first target lexical token at $i = 1$. This state is not yet related to the source representations. The upper decoder layers establish the connection to the encoder, each of which has two attention sublayers: self-attention and encoder-decoder attention.

Self-Attention. The first step of a decoder layer is to contextualize the states of the previous layer up to position i. This is done with self-attention but slightly differently from that of an encoder:

$$\mathbf{q}_i^{(l)} = \sum_{i'=1}^{i} \alpha_{ii'}^{(l)} \cdot \mathbf{s}_{i'}^{(l-1)} \tag{3.26}$$

where the weights are normalized over the past target positions (*left-sided attention*):

$$\begin{align} \alpha_{ii'}^{(l)} &= \operatorname{softmax}(\mathbf{s}_i^{(l-1)\top} \cdot \mathbf{S}_i^{(l-1)})_{i'} \tag{3.27} \\ &= \frac{\exp(\mathbf{s}_i^{(l-1)\top} \cdot \mathbf{s}_{i'}^{(l-1)})}{\sum\limits_{i''=1}^{i} \exp(\mathbf{s}_i^{(l-1)\top} \cdot \mathbf{s}_{i''}^{(l-1)})} \tag{3.28} \end{align}$$

where $\mathbf{S}_i^{(l)} \in \mathbb{R}^{D_h \times i}$ is a concatenation of all states of layer l at positions from 1 to i. It is unavoidable to constrain the attention like this since a decoder does not yet have the states of the future positions where the target words are not predicted.

Encoder-Decoder Attention. The second sublayer actually identifies which parts of the source sentence are relevant for predicting the current target word. Technically, given the self-attended representation $\mathbf{q}_i^{(l)}$ as a summary of the prediction process so far, encoder-decoder attention calculates the weight of each encoder representation $\mathbf{h}_j$ with respect to $\mathbf{q}_i^{(l)}$:

$$\begin{align} \alpha_{ij}^{(l)} &= \operatorname{softmax}(\mathbf{q}_i^{(l)\top} \cdot \mathbf{H})_j \tag{3.29} \\ &= \frac{\exp(\mathbf{q}_i^{(l)\top} \cdot \mathbf{h}_j)}{\sum_j \exp(\mathbf{q}_i^{(l)\top} \cdot \mathbf{h}_j)} \tag{3.30} \end{align}$$

The state of the decoder layer l is a weighted sum of source representations $\mathbf{h}_1^J$:

$$\mathbf{s}_i^{(l)} = \sum_{j=1}^{J} \alpha_{ij}^{(l)} \cdot \mathbf{h}_j \tag{3.31}$$

In the form of query/key/value matrices, it can be written as follows:

$$\mathbf{S}_i^{(l)} = \operatorname{softmax}(\mathbf{Q}_{e,i}^{(l)\top} \cdot \mathbf{K}_e) \cdot \mathbf{V}_e \tag{3.32}$$

where $\mathbf{Q}_{e,i}^{(l)} \in \mathbb{R}^{D_h \times i}$ and $\mathbf{K}_e = \mathbf{V}_e = \mathbf{H}$. Note that self-attention of a decoder layer builds a query for the encoder-decoder attention of the same layer.

The encoder-decoder attention weight (Equation 3.30) can be compared with the word alignment of traditional MT (Equation 3.4). Both concepts associate source positions with target positions, but there are the following technical differences:

- The attention weights compute soft alignments, i.e. nonzero probabilities over all positions on the opposite side, while the traditional word alignments are hard, discrete decisions.
- The attention weights are used to aggregate source representations, which directly conditions the prediction of target words. On the other hand, the traditional word alignments are used to train reordering features which give partial scores when predicting a target word.
- The attention weights are integrated into the translation model and trained jointly with other parts of the model, while the traditional word alignments are obtained from a completely independent model.

- The attention weights may be computed in a chain over multiple layers within the same model. The traditional word alignments may be computed differently with different alignment models.

All in all, the encoder-decoder attention weight cannot exactly correspond to the traditional word alignment, and is very blurry and often not interpretable [Peter & Nix$^+$ 17]. Nonetheless, the attention components offer more flexible and united modeling for NMT, which usually leads to a stronger performance than phrase-based MT [Bentivogli & Bisazza$^+$ 16].

The two attention operations are repeated alternatingly over L_e layers. Finally, the ultimate decoder representation is given by the output of the last decoder layer, i.e. $\mathbf{s}_i = \mathbf{s}_i^{(L_e)}$.

Output Layer

The decoder indeed compresses the context of the target generation history (e_1^{i-1}) and the source sentence (f_1^J) into a single vector $\mathbf{s}_i$. This representation is used to compute the probability of generating a target word $e_i \in \mathcal{V}_e$ by first applying a linear transformation, followed by a softmax function over all target vocabulary entries:

$$p(e_i|e_0^{i-1}, f_1^J) = \mathrm{softmax}(\mathbf{W}_o \mathbf{s}_i + \mathbf{b}_o)_i \tag{3.33}$$

where $\mathbf{W}_o \in \mathbb{R}^{|\mathcal{V}_e| \times D_h}$ is the weight and $\mathbf{b}_o \in \mathbb{R}^{|\mathcal{V}_e|}$ is the bias of the output layer. Note that the resulting probability distribution is also stored as a vector of $|\mathcal{V}_e|$ dimensions, each of which corresponds to the integer index of a vocabulary entry in $\mathcal{V}_e = \{1, ..., |\mathcal{V}_e|\}$.

Since the vocabulary size $|\mathcal{V}_e|$ is huge in natural languages, e.g. hundreds of thousands, normalizing over the vocabulary is very expensive in the softmax function and often dominates the complexity of an NMT model. This can be speeded up by limiting the vocabulary to a fixed number of most frequent words [Sutskever & Vinyals$^+$ 14] or using a more fine-grained vocabulary, e.g. subwords [Sennrich & Haddow$^+$ 16c] or characters [Lee & Cho$^+$ 17].

Given the distribution of Equation 3.33, we may either take the maximally probable entry or a set of most probable entries in the target vocabulary. The chosen entry is set as the hypothesis e_i and extends the target generation history from e_1^{i-1} to e_1^i. This is fed back as the input to the decoder, renewing the query for encoder-decoder attentions at position $i+1$. The details of these choices and constructing target hypothesis sequences are explained in Section 3.3.3.

Transformer Architecture

Thus far, we have examined the core components of an attention-based sequence-to-sequence NMT model. These are also the most linguistically intuitive parts in implementing a translation process as a neural network, and a fully working NMT model can be built just with these components. From here on, we introduce some variants of each component or additional concepts that might be less linguistic but rather mathematical, but which, however, make the computational modeling of MT more effective. These complete the original Transformer architecture [Vaswani & Shazeer$^+$ 17] on which all methods in this thesis are based (Figure 3.1).

Multi-Head Attention. For each layer, an attention component computes a single set of weights for values. Multi-head attention extends this to learn multiple sets of weights, each of which attends to a subspace of value representations. An index of each weight set, correspondingly each subspace, is referred to as a *head*. This enables the attention to focus on input positions differently according to diverse criteria [Voita & Talbot$^+$ 19].

For example, let us reformulate the encoder-decoder attention (Equation 3.29–3.32) with Z heads. First of all, the query vector $\mathbf{q}_i^{(l)}$ and encoder representation $\mathbf{h}_j$ are mapped to a subspace

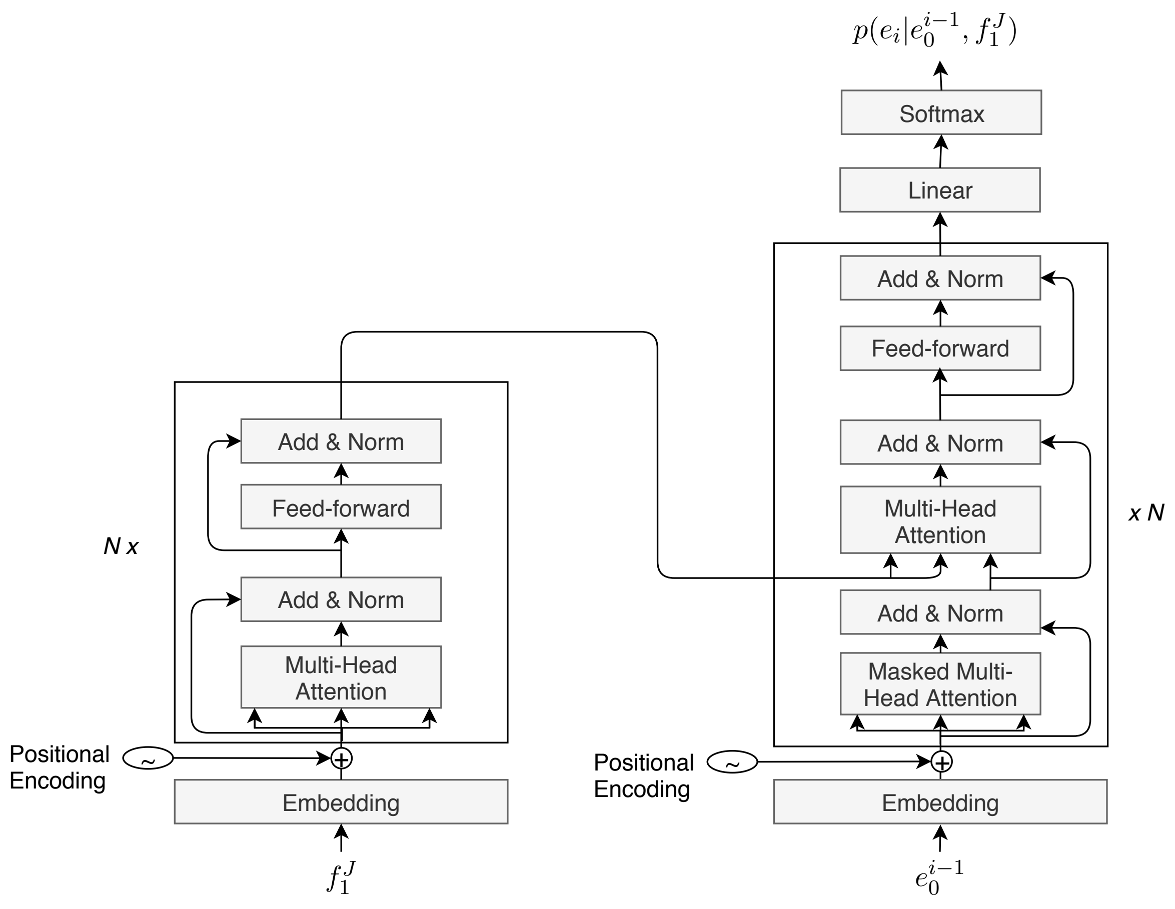


Figure 3.1: Transformer model architecture for NMT. Adapted from [Petrov 19].

of each head z:

$$\mathbf{q}_{iz}^{(l)} = \mathbf{W}_{qz}^{(l)} \cdot \mathbf{q}_i^{(l)} \tag{3.34}$$

$$\mathbf{h}_{jz}^{(l)} = \mathbf{W}_{hz}^{(l)} \cdot \mathbf{h}_j \tag{3.35}$$

where $\mathbf{W}_{qz}^{(l)}, \mathbf{W}_{hz}^{(l)} \in \mathbb{R}^{(D_h/Z)\times D_h}$ are trainable parameters. The attention weights are computed for each z individually:

$$\alpha_{zij}^{(l)} = \frac{\exp(\mathbf{q}_{iz}^{(l)\top} \cdot \mathbf{h}_{jz}^{(l)})}{\sum_{j=1}^{J} \exp(\mathbf{q}_{iz}^{(l)\top} \cdot \mathbf{h}_{jz}^{(l)})} \tag{3.36}$$

$$\mathbf{s}_{iz}^{(l)} = \sum_{j=1}^{J} \alpha_{zij}^{(l)} \cdot \mathbf{h}_{jz} \tag{3.37}$$

Then it concatenates $\mathbf{s}_{iz}^{(l)}$ over all heads and applies the last linear projection with $\mathbf{W}_s^{(l)} \in \mathbb{R}^{D_h \times D_h}$:

$$\mathbf{s}_i^{(l)} = \left[\mathbf{s}_{iz}^{(l)} \,\middle|\, \forall z \in \{1, ..., Z\} \right] \cdot \mathbf{W}_s^{(l)} \tag{3.38}$$

Self-attention layers are modified analogously.

Dot Product Scaling. Attention weights in Transformer (α) are calculated by applying the softmax function to dot products of queries and keys. The magnitude and variance of the dot products get larger for high-dimensional queries/keys, which makes the softmax output very peaked and the corresponding gradients extremely small. To counteract this effect, Transformer scales the dot products for all attention components by $1/\sqrt{D_h}$. In the case of the encoder-decoder attention (Equation 3.29), this results in:

$$\alpha_{ij}^{(l)} = \text{softmax}\left(\frac{\mathbf{q}_i^{(l)\top} \cdot \mathbf{H}}{\sqrt{D_h}}\right)_j \tag{3.39}$$

Vocabulary/Embedding Sharing. If source and target languages are linguistically similar with (partially) shared alphabets, it makes sense to have a single joint vocabulary between the two languages. This is often built at the subword level [Sennrich & Haddow$^+$ 16c] to increase the number of shared lexical tokens and to learn a denser distribution that helps to cope with rare words. Accordingly, the source embedding weight can be tied with the target embedding weight, i.e. $\mathbf{W}_f = \mathbf{W}_e$. The sharing of a vocabulary and embedding has these advantages:

- Lexical knowledge is directly shared via embeddings of common vocabulary entries.
- The model size gets effectively smaller and the softmax computations become faster.
- It provides regularization by training the same parameter with different connections within the model.

Similarly, the target embedding weight can be coupled with the output layer weight, i.e. $\mathbf{W}_e = \mathbf{W}_o^\top$ [Press & Wolf 17]. Those two weight matrices are also called input and output embeddings in the context of language modeling, both of which are extracted as features for other NLP applications [Mikolov & Chen$^+$ 13, Mnih & Kavukcuoglu 13, Nalisnick & Mitra$^+$ 16]. They are transposes of each other in shape and symmetric in behavior. Each column of the target embedding weight stores a representation vector for each target vocabulary entry; each row of the output layer weight converts a representation vector to a score of the corresponding target word. Note that the softmax output of these scores converges to a one-hot vector—the form of the input to the target embedding—when the prediction becomes extremely certain. For semantically similar target words, the goal of the target embedding weight is having similar representations, namely similar columns. In the output layer weight, it is desired to have similar rows for them, producing similar scores for the softmax [Mnih & Teh 12]. Therefore, it is reasonable to treat the two weights as one and train them jointly.

In addition, Transformer multiplies source and target embeddings by $\sqrt{D_h}$. For these input embedding vectors, the lengths should be sufficiently large to ensure the word embedding vectors are lexically distinguishable. On the other hand, if the output layer weight yields too large magnitudes, the following softmax gets only negligible gradients, just as the attention case. For the input-output embedding sharing, the scaling factor on the input side assures large magnitudes of the input embedding vectors, while keeping the shared weight reasonably small for meaningful gradients in the output layer.

Transformer often ties all three weights of source/target embeddings and output layer.

Feedforward Sublayer. At the end of each encoder/decoder layer, Transformer inserts an additional feedforward unit of two linear transformations with a rectifier linear unit (ReLU) activation in between, e.g.

$$\mathbf{s}_i^{(l)} = \mathbf{W}_{\text{ff},2}^{(l)} \cdot \text{ReLU}\left(\mathbf{W}_{\text{ff},1}^{(l)} \cdot \bar{\mathbf{s}}_i^{(l)} + \mathbf{b}_{\text{ff},1}^{(l)}\right) + \mathbf{b}_{\text{ff},2}^{(l)} \tag{3.40}$$

with Equation 3.31 being renamed to $\bar{\mathbf{s}}_i^{(l)}$. This incorporates non-linearity into the model explicitly and strengthens the model capacity. A feedforward sublayer has four additional parameters $\mathbf{W}_{w,1}^{(l)} \in \mathbb{R}^{D_w \times D_h}$, $\mathbf{b}_{w,1}^{(l)} \in \mathbb{R}^{D_w}$, $\mathbf{W}_{w,2}^{(l)} \in \mathbb{R}^{D_h \times D_w}$, and $\mathbf{b}_{w,2}^{(l)} \in \mathbb{R}^{D_h}$, where D_w controls the increase in model size and is normally larger than D_h.

Residual Connection & Layer Normalization. To improve the stability of training, Transformer employs residual connection [He & Zhang+ 16] and layer normalization [Ba & Kiros+ 16] for each sublayer. In the example of the feedforward sublayer in a decoder layer, Equation 3.40 is modifed to:

$$\mathbf{s}_i^{(l)} = \text{layernorm}\left(\bar{\mathbf{s}}_i^{(l)} + \mathbf{W}_{w,2}^{(l)} \cdot \text{ReLU}\left(\mathbf{W}_{w,1}^{(l)} \cdot \bar{\mathbf{s}}_i^{(l)} + \mathbf{b}_{w,1}^{(l)}\right) + \mathbf{b}_{w,2}^{(l)}\right) \tag{3.41}$$

3.3.2 Training

Training of a computational model means finding a set of appropriate values for the model parameters. For any model, it universally requires three components to be prepared:

- A mathematical function to be maximized (*score*) or minimized (*loss*) with respect to the model parameters: *training criterion.*
- A computational algorithm to optimize the training criterion: *optimization algorithm.*
- Heuristic techniques to complement the optimization problem for controlling the final performance: *regularization.*

For NMT, there is basically only one encoder-decoder model for $p(e_i|e_1^{i-1}, f_1^J)$, for which we need to fill the three requirements for training.

Training Criterion

Defining a training criterion involves this question: What would be a set of appropriate values for the model parameters, and which function should we optimize to obtain such parameter values? Statistical approaches design the criteria so that the model learns to explain given training data well. An MT system is supposed to precisely model a bilingual training corpus, i.e. the relation between source and target sentences in it.

NMT models are mostly trained with the cross-entropy loss of the model p_θ relative to the true bilingual sentence distribution $Pr(f_1^J, e_1^I)$:

$$\begin{aligned} \mathrm{L}(\theta) &= -\sum_{(f_1^J, e_1^I)} Pr(f_1^J, e_1^I) \log p_\theta(e_0^I | f_1^J) &\quad (3.42)\\ &= -\frac{1}{|\mathcal{C}_{f,e}|} \sum_{(f_1^J, e_1^I) \in \mathcal{C}_{f,e}} \log p_\theta(e_0^I | f_1^J) &\quad (3.43)\\ &= -\frac{1}{|\mathcal{C}_{f,e}|} \sum_{(f_1^J, e_1^I) \in \mathcal{C}_{f,e}} \sum_{i=1}^{I} \log p_\theta\big(e_i | e_0^{i-1}, f_1^J\big) &\quad (3.44) \end{aligned}$$

where θ is a set of model parameters. The true distribution is approximated by an empirical distribution over given bilingual training data $\mathcal{C}_{f,e}$. In the end, we optimize the logarithm of the model probability itself, iterating over the training data. The cross-entropy encourages the probability of the correct target word e_i to increase, while suppressing the probability of all other target vocabulary words in $\mathcal{V}_e$. For each target position i, the log probability in Equation 3.44 turns into:

$$\log p_\theta\big(e_i | e_0^{i-1}, f_1^J\big) = \log(\text{softmax}(\mathbf{o}_i)_{e_i}) \tag{3.45}$$

$$= \mathbf{o}_{ie_i} - \log \sum_{e \in \mathcal{V}_e} \exp(\mathbf{o}_{ie}) \tag{3.46}$$

where $\mathbf{o}_i$ is the logit score vector before the softmax activation:

$$\mathbf{o}_i = \mathbf{W}_o \mathbf{s}_i + \mathbf{b}_o \tag{3.47}$$

In Equation 3.46, the first term has a direct contribution to the loss, leading to maximizing the score of e_i. The second term forces the score sum to be minimized by pushing all other scores for $e \neq e_i$ down.

Note that the model probability of e_i depends on the target history e_1^{i-1}. In training, this history is drawn from the target side of the training data with the future tokens (e_i^I) masked out. In contrast, in testing, the history is generated by the model itself and extended by one token at a time as the decoding predicts more tokens. This is a well-known discrepancy between the training and testing of NMT models, called *exposure bias* [Ranzato & Chopra$^+$ 16].

Optimization Algorithm

Optimization of Equation 3.44 has no closed-form solution; it is therefore done with a numerical gradient-based algorithm.

Forward Pass. The first step of the optimization is to calculate the actual loss value $\mathrm{L}(\theta)$. Using the current set of parameters, we feed the training data to the model and perform the network computations from bottom to top until the final output layer. In neural network literature, this flow is called a *forward* pass, as opposed to the backward flow of gradients that will be explained below. Having estimated the model outputs, we perform additional operations with regard to the gold outputs, i.e. the target reference sentence e_1^I, to obtain the final loss value. For the cross-entropy loss, we only need to choose the corresponding entry for e_i in the softmax probability vectors.

Backpropagation. Next, we take a derivative of the computed loss with respect to each parameter of the network. An NMT model is a complex chain of linear/non-linear functions; the gradients are *propagated back* from the top (output layer) to the bottom via the chain rule of differentiation [Rumelhart & Hinton$^+$ 86]. For example, the gradient with respect to the weight matrix in the output layer ($\mathbf{W}_o$) is computed as follows:

$$\frac{\partial \mathrm{L}}{\partial \mathbf{W}_o} = \frac{\partial \mathrm{L}}{\partial \mathbf{o}} \cdot \frac{\partial \mathbf{o}}{\partial \mathbf{W}_o} \tag{3.48}$$

where L is a function of $\mathbf{o}$ and $\mathbf{o}$ is a function of $\mathbf{W}_o$, according to Equation 3.46. For the lower layers, we need to expand the product of gradients further through the dependency chain of $\mathbf{s}_i$ (Equation 3.47). The procedure continues until the bottommost (input) layer of the network.

Gradient Descent. Once all the gradients have been computed, each parameter W of the network is updated along the direction of the gradient:

$$\tilde{\mathbf{W}} := \mathbf{W} - \eta \cdot \frac{\partial \mathrm{L}}{\partial \mathbf{W}} \tag{3.49}$$

where $\tilde{\mathbf{W}}$ is the new parameter value after the update. The cycle of forward pass, backpropagation, and gradient update is repeated until a stopping criterion is met. This series of gradual updates is called *gradient descent*, which heads to the minimum of a loss function. When the training criterion is a score to be maximized, the sign of the update in Equation 3.49 should be flipped (*gradient ascent*).

η is the *learning rate* which controls the amount of change. If it is too small, the parameters are shifted so slowly that it might take a long time to converge to a good optimum. A large learning rate can be harmful near the optimum, making the parameter value skip back and forth around the optimum and diverge from it. Hence, it is important to set an appropriate value for the learning rate according to how learning is progressing. The learning rate adjustment can be done with a manually defined schedule or an automatic algorithm using the statistics of the erstwhile updates [Duchi & Hazan$^+$ 11, Zeiler 12, Kingma & Ba 15].

Mini-Batching. As the training data gets larger, it is memory-intensive to process the whole data for each update (Equation 3.44). Also, if the updates are done only at the end of iterating the entire corpus, we cannot monitor the intermediate progress of training and thus it takes a long time to determine whether to stop the training. The other extreme is *stochastic gradient descent* (SGD), where the parameters are updated for each example in the training data. This enables the model to react faster in the optimization but yields too high variance in the loss and its gradients, which makes the training unstable. Either way has potential to prolong the overall training time and miss good local optima.

Mini-batching is a scheme in between; it splits the data into small portions and updates the parameters after each portion is processed. Each portion is called a *mini-batch*, and the *batch size* is normally around dozens or hundreds of sentences in NMT, which is much smaller than the size of typical bilingual training corpora, e.g. millions of sentences. This way, the network updates are more frequent than the whole-corpus updates, which tends to find better local minima more sensitively. It is not as frequent as in the stochastic variant, so the optimization is less affected by noisy training examples. Furthermore, it is computationally more efficient in terms of memory and convergence speed.

For efficient parallel processing of multiple sentences, a mini-batch is constructed as a matrix with the size as follows: number of sentences in the mini-batch (batch size) $\times$ maximum sentence length in the mini-batch. Sentences that are shorter than the maximum length are appended with pre-defined *padding* tokens to fill up the matrix. This input matrix is fed to the model network, which can be implemented solely with matrix-vector calculations using modern neural network libraries [Chen & Li$^+$ 15, Abadi & Barham$^+$ 16, Paszke & Gross$^+$ 19]. In this way, the forward and backward passes are highly efficient in a graphic processor unit (GPU).

One trick to increase the throughput is *bucketing*, which sorts the sentences by length so that each mini-batch contains sentences of similar lengths. This minimizes the matrix size by reducing the redundant padding, and also stabilizes the training by letting the model learn short and simple examples first and gradually adapt to more complex and longer examples. [Bengio 12] contains more details on practical guidelines for mini-batching.

Regularization

Fully optimizing a training criterion is intended, but it might lead to *overfitting*: reaching a bad local optimum which explains only the given training data well and does not generalize to unseen examples. Overfitting often happens in a low-resource scenario when the given data is too small to cover the whole test domain. Regularization is an additional measure to tweak the pure optimization in a way that avoids overfitting. In this section, we present three common regularization methods for NMT, which adjust the training procedure, training data, and the model, respectively.

Early Stopping. An intuitive way to circumvent over-optimization is just to stop the training in the middle, even while the loss value is still decreasing. However, if we cease the training too early, the model might not be trained sufficiently and represent neither the training data nor the testing domain. In order to find a satisfying optimum for both, one can check the model's performance periodically on a small corpus which is distinct from the training corpus. Such a corpus is called a *development set*, which resembles the test domain but does not overlap with the final test sets. The development set performance indicates how well the model generalizes to the test domain, while the actual test sets are not exposed to the model during training. The performance is usually measured with *perplexity*:

$$\mathrm{PPL}(p_\theta, \mathcal{C}^{\mathrm{d}}_{f,e}) = \left(\prod_{(f_1^J, e_1^I) \in \mathcal{C}^{\mathrm{d}}_{f,e}} \prod_{i=1}^{I} p_\theta(e_i | e_0^{i-1}, f_1^J) \right)^{-\frac{1}{\sum\limits_{(f_1^J, e_1^I) \in \mathcal{C}^{\mathrm{d}}_{f,e}} I}} \tag{3.50}$$

where $\mathcal{C}^{\mathrm{dev}}_{f,e}$ is a development set. Perplexity is an inverse geometric average of model probabilities over the target positions, which is basically an exponentiated form of the cross-entropy (Equation 3.44). This indicates how many target vocabulary entries can be practically chosen per word position. Otherwise, translation metrics (Section 3.6.3) are also used to check the intermediate progress of training.

A *checkpoint* means the point when the development set performance is validated. The interval between checkpoints (*checkpoint frequency*) is usually specified by the number of parameter updates conducted in it, i.e. mini-batches. Typically, at each checkpoint, the learning rate is also adjusted along the pre-defined rules.

Label Smoothing. Another way is to augment training data with soft labels, i.e. perturbations of the reference translations, considering that the given data is finite and not perfect. The idea is that we do not totally trust the given data but allow different translations of each source sentence, which makes the model more flexible to cope better with unseen test examples.

Label smoothing [Szegedy & Vanhoucke+ 16] is a simple, systematic method to implement this idea without any further knowledge about the data. It redistributes a part of the probability mass of each word in the reference to all other words in the target vocabulary, modifying the training criterion (Equation 3.44) as follows:

$$\mathrm{L}(\theta) = -\frac{1}{|\mathcal{C}_{f,e}|} \sum_{(f_1^J, e_1^I) \in \mathcal{C}_{f,e}} \sum_{i=1}^{I} \sum_{e \in \mathcal{V}_e} \left[\epsilon \cdot \frac{1}{|\mathcal{V}_e|} + (1-\epsilon)\delta_{e,e_i} \right] \cdot \log p_\theta\big(e | e_0^{i-1}, f_1^J\big) \tag{3.51}$$

where ϵ is the amount of probability mass to be redistributed, with one as the original mass of the reference word. For example, $\epsilon = 0.1$ means that we trust the reference word with 90% confidence and leave room for other words with the other 10%.

Dropout. Alternatively, we may make temporary alterations to the model during the training so that not every parameter is fully optimized to the training criterion. Dropout is such a technique specifically designed for neural networks, which randomly zeros out some dimensions from a layer output [Srivastava & Hinton+ 14]. This effectively drops the corresponding rows of the layer weight from backpropagation, while maintaining the overall architecture of the network. By randomizing the dropped dimensions for each training instance, it approximates a bagging ensemble [Dietterich 00] of many randomly masked networks. Dropout can be selectively applied to any layer of a network, typically with the same zeroing-out rate for those layers. The technique is not restricted to a specific network architecture. After training, dropout is turned off at inference time.

3.3.3 Decoding

Once the model $p(e_i | e_1^{i-1}, f_1^J)$ has been trained, decoding produces an actual translation of the given source sentence f_1^J by searching for a sequence e_1^I that has the highest model score combined over the length I (Equation 3.15). Note that the search space is intractable; for each target position, we have $|\mathcal{V}_e|$ choices for the output, which is usually in the order of thousands to millions. Since we do not know I beforehand, the search complexity can amount to $O(|\mathcal{V}_e|^{I_\mathrm{m}})$ with I_m as the predefined maximal length, e.g. a hundred. Hence, a reasonable approximation on the search space is key to making a decoding algorithm practical.

Beam Search

Beam search is the dominating algorithm for an efficient decoding with the NMT model of Section 3.3.1. It regards the target hypothesis space as a directed graph from left to right on target positions, complying naturally with the factorization of the model (Equation 3.15). A path in the graph represents a (partial) hypothesis, where each edge means an expansion of a hypothesis with a next target token. Each node contains the model score and the pre-computed network states of the hypothesis, which are updated cumulatively to avoid redundant computations. The idea is to prune unreliable hypotheses in the middle of traversing a search graph, according to the partial score for the traversed path.

1. Let the root node be the initial hypothesis with only the sentence start symbol ($e_0 =$ `<s>`). Set its score to zero and perform the forward pass of $p(e|e_0, f_1^J)$.
2. Expand the initial hypothesis with the top-N target words $e \in \mathcal{V}_e$ having the highest probabilities $p(e|e_0, f_1^J)$. For each expansion $e_1 = e$, update the model score to $\log p(e|e_0, f_1^J)$.
3. Keep only the N best paths (*beam*) with the highest scores and drop the other hypotheses (free the memory).
4. Expand each node in the beam again with the top-N target words at the next position i, generating N^2 path s. For each expansion $e_i = e$, add $\log p(e|e_0^{i-1}, f_1^J)$ to the hypothesis score.
5. Keep only the N best paths. Once a path has hypothesized the sentence end symbol (`</s>`), it should be not expanded further and is kept in the beam.
6. Repeat steps 4–5 until all hypotheses in the beam reach either the sentence end or $i = I_\mathrm{m}$.

N is the *beam size*; at the end of the procedure for every position, only N (partial) hypotheses remain in the beam during decoding. The beam search does not guarantee to find the global optimum, but reduce the search complexity to $O(N \cdot I_\mathrm{m})$. One can trade the performance off against the decoding speed by adjusting the beam size. If $N = 1$, we keep track of only one locally best path, which is called *greedy search.* Note that hypotheses are evaluated in batch mode, i.e. for each position, the translation model performs the forward pass for multiple hypotheses with a single call to the computation device.

Length Control

Another concern for NMT decoding is that the decision rule (Equation 3.14–3.15) fundamentally prefers shorter hypotheses. The probability $p(e_1^I|f_1^J)$ decreases with increasing target length I; if the search algorithm hypothesizes the sentence end symbol quite early and keeps the short hypothesis in the beam, it is very likely to remain as the highest scoring path at the end. This problem cannot be solved just by learning to put a high probability for the target reference sentence with an appropriate length (Equation 3.44). Thus the NMT usually resorts to additional factors in the decision rule, which guide the decoding algorithm only to the hypotheses with reasonable lengths. A heuristic to counteract the length bias is normalizing the score by its target length I:

$$f_1^J \mapsto \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I,\, e_1^I}{\operatorname{argmax}} \left\{ \left[\prod_{i=1}^{I} p(e_i|e_0^{i-1}, f_1^J) \right]^{\frac{1}{I^\omega}} \right\} \tag{3.52}$$

$$= \underset{I,\, e_1^I}{\operatorname{argmax}} \left\{ \frac{1}{I^\omega} \sum_{i=1}^{I} \log p(e_i|e_0^{i-1}, f_1^J) \right\} \tag{3.53}$$

where $\omega \in \mathbb{R}$ (*length penalty*) gives an additional control of the length [Wu & Schuster$^+$ 16]. Increasing ω makes the decoding prefer longer hypotheses. The normalization is applied in reranking hypotheses in the beam after the search procedure has finished. During the expansion steps, all non-final hypotheses have the same length, so the normalization is not necessary.

3.4 Language Modeling

This section describes the details of language modeling, which has been a crucial model component in MT. It is responsible for the fluency of the target output in the word-based noisy channel (Section 3.2.1) and phrase-based log-linear model (Section 3.2.2). In the main contents of this thesis, an LM is actively exploited in semi-supervised learning (Section 4.2 and 4.4) and unsupervised learning (Section 6.2 and 6.3), as it is a simple yet strong model of sentence structures that can be trained with only monolingual data.

The task of language modeling is defined as estimating the probability of a sequence of words $Pr(e_0^I)$, decomposed over the word positions:

$$Pr(e_0^I) = Pr(e_I|e_1, e_2, ..., e_{I-1}) \cdot Pr(e_{I-1}|e_1, e_2, ..., e_{I-2}) \cdots Pr(e_1|e_0) \cdot Pr(e_0) \tag{3.54}$$

$$= \prod_{i=1}^{I} Pr(e_i|e_0^{i-1}) \tag{3.55}$$

where $Pr(e_0) := 1$. For simplicity, we model the decomposed term $Pr(e_i|e_0^{i-1})$ instead of directly modeling $Pr(e_0^I)$. Each term corresponds to predicting the next word given a history of words, from left to right. The product of the terms gives the sentence probability with mathematically correct normalization. This provides an idea of how good a sentence is structured in the desired language, implicitly modeling the syntax and fluency. Specifically, the probability provided by an LM helps to decide between two similar words, where one is more likely to appear in the context of the sentence than the other. As mentioned before, an LM can be built solely with monolingual text, which does not require additional labeling; each word plays a role as a label by itself for the prediction task at each position.

An LM $p_\theta(e_i|e_0^{i-1})$ is trained using the maximum likelihood criterion, or equivalently, minimizing the negative log-likelihood:

$$L(\theta) = -\frac{1}{|\mathcal{C}_e|} \sum_{e_1^I \in \mathcal{C}_e} \sum_{i=1}^{I} \log p_\theta(e_i|e_0^{i-1}) \tag{3.56}$$

where $\mathcal{C}_e$ is the monolingual training corpus. The performance of an LM is evaluated by its perplexity on a test corpus $\mathcal{C}_e^{\text{test}}$:

$$\text{PPL}(p_\theta, \mathcal{C}_e^{\text{t}}) = \left(\prod_{e_1^I \in \mathcal{C}_e^{\text{t}}} \prod_{i=1}^{I} p_\theta(e_i|e_1^{i-1}) \right)^{-\frac{1}{\sum_{e_1^I \in \mathcal{C}_e^{\text{t}}} I}} \tag{3.57}$$

The training criterion and the perplexity definition are analogous to those of NMT (Section 3.3.2).

In the following, we explain two types of LMs, depending on the model assumption and architecture: count-based and neural.

3.4.1 Count-based Language Model

The initial approach to language modeling was to use the counts of word-history pairs. These counts are computed from the relative frequencies of word sequences in the training data, which is the most primitive method for the maximum likelihood estimation. The number of counts to store might grow exponentially by vocabulary size, if we consider arbitrary sequence lengths in training. Therefore, an intuitive assumption is made by the count-based approach, trimming long histories and retaining at most $n-1$ previous words in a history:

$$\begin{aligned} Pr(e_0^I) &= \prod_{i=1}^{I} Pr(e_i|e_{i-(n-1)}^{i-1}) \qquad (3.58) \\ &= p_\theta(e_i|e_{i-n+1}^{i-1}) \qquad (3.59) \end{aligned}$$

A sequence of a current word and the $n-1$ previous words is called an n-gram. It is reasonable to truncate a history since a word is usually less related to distant words in a sentence. By restricting a model to n-gram events, we can guarantee that the count table is small enough to fit in memory with $O(|\mathcal{V}_e|^n)$. Throughout this thesis, interpolated modified Kneser-Ney smoothing [Chen & Goodman 99] was used to train the n-gram models. This smoothing method spreads the probability mass of n-grams to $(n-1)$-grams, preventing a zero probability of unseen n-grams.

3.4.2 Neural Language Model

A count-based LM is a straightforward concept, but its n-gram structure sacrifices the information from long-range contexts. As in Section 3.3, some neural network architectures enable an LM to handle long contexts without the shortening assumption on the history length:

$$Pr(e_i|e_1^{i-1}) = p_\theta(e_i|e_1^{i-1}) \qquad (3.60)$$

given a reasonable amount of free parameters. An LM with an unlimited context is realized by either recurrent or self-attention units, which resembles the decoder of an NMT model (Section 3.3.1). As an example, a self-attentional LM has the same architecture as a self-attentional NMT decoder except that it is not conditioned on a source sentence, i.e. has no encoder-decoder attentions:

$$\mathbf{s}_i^{(l)} = \mathbf{q}_i^{(l)} \qquad (3.61)$$

where the output of each layer is just the self-attention result (c.f. Equation 3.31). Training and inference of a neural LM are also analogous to that of an NMT model, where an output sentence is the same as its input yet with the words shifted by a position to the left. Note that, to get scores for all words in the vocabulary, a neural LM computes all of them in a single forward step, while a count-based LM needs to be queried for each word in the vocabulary individually. Neural LMs outperform count-based LMs thanks to their more expressive modeling in continuous-valued spaces with implicit smoothing [Bengio & Ducharme$^+$ 03, Mikolov & Karafiát$^+$ 10].

3.5 Cloze Task Modeling

Recently, neural LMs have also been used to obtain useful representations that can be utilized later for other NLP tasks [Radford & Narasimhan$^+$ 18, Peters & Neumann$^+$ 18]. In particular, one is interested in extracting layer states $\mathbf{s}_i^{(l)}$ instead of conditional probabilities of words given a history. For this purpose, a neural network should not necessarily be trained to predict a word given its previous words. The dependence on a previous history is derived from the left-to-right decomposition of a sentence probability (Equation 3.54). However, if our interest is not a sentence probability but just a representation vector for other usages, there is no need to train a model to compute the left-conditioned probabilities. One may devise a task that is more effective in learning representations, which is possibly more complex than language modeling.

The Cloze task [Taylor 53] is such an alternative that guesses a randomly deleted word in a sentence. If a word is deleted at position i, a model for this task will be:

$$p(e_i|e_1, ..., e_{i-1}, \texttt{<m>}, e_{i+1}, ..., e_I) \tag{3.62}$$

where the word is masked out with a special token `<m>`. The model still has access to all other words in the sentence and their sequential order, which are used to predict the original word e_i at position i. Note that, in contrast to an LM whose prediction depends only on the left-side context, this model considers the left and right contexts together. [Devlin & Chang$^+$ 19] extend this process with more options in the masking. Instead of putting a fixed token `<m>`, one can replace it with another real word:

$$p(e_i|e_1, ..., e_{i-1}, e^*, e_{i+1}, ..., e_I) \tag{3.63}$$

where $e^* \in \mathcal{V}_e$ is chosen randomly from the vocabulary. In this case, the model should detect that the word e^* is irrelevant to the context and ignore its influence in predicting the output. Another option is to keep the original word e_i:

$$p(e_i|e_1, ..., e_{i-1}, e_i, e_{i+1}, ..., e_I) \tag{3.64}$$

where the model is supposed to copy the input to the output at position i. In the following, we formulate the details of designing and training a model for these tasks and how to use their representations afterwards, based on [Devlin & Chang$^+$ 19].

3.5.1 Model

For efficiency and knowledge transfer (see the introduction of Chapter 5), we use the same model shared among all tasks of Equations 3.62-3.64 and arbitrary word positions. Such a model should process the input sentence for all masked/unmasked positions at the same time. A self-attentional encoder (Section 3.3.1) fits nicely to this purpose as it learns representations from bidirectional contexts and its forward step of every position can be run in parallel. Here, for the Cloze task modeling, we attach an output layer on top of the encoder, removing the unidirectional decoder. Note that the output prediction at each position is not dependent on the decisions at other positions. Unlike an LM, this model does not guarantee normalization at the sentence level when the probabilities for all positions are multiplied together.

3.5.2 Training

It is ideal to train a model with the three corrupting options for all positions of every sentence; however, this would be however too slow with a large training corpus. Hence, the training is done in a stochastic way, randomly selecting the position and the corruption:

1. Uniformly sample a position to be corrupted from a sentence. Repeat until the number of such positions reaches $\lceil \rho \cdot I \rceil$, where $\rho \in [0, 1]$ is a hyperparameter to control the number of corruptions per sentence.

2. Randomly choose the type of corruption with a probability ρ_m for masking with `<m>`, ρ_r for replacing with a random word, ρ_k for keeping the original word, where $\rho_\mathrm{m}, \rho_\mathrm{r}, \rho_\mathrm{k} \in [0, 1]$ and $\rho_\mathrm{m} + \rho_\mathrm{r} + \rho_\mathrm{k} = 1$.
3. Perform the corrupting operation. If the random word corruption is chosen, sample a word e^* from the vocabulary $\mathcal{V}_e \setminus \{e_i\}$ and replace the original with it.

This stochastic corruption is done on the fly during training so that the model sees a different task for the same sentence for each epoch. The training criterion is the cross-entropy (Equation 3.44) but only on the corrupted positions that are selected in Step 1. The model still processes all other positions with intact words and learns to produce their representations, from which the original of the corrupted positions are inferred.

Table 3.1 shows an example of the input corruption and its training. In the sentence e_1^I = "Thanks for reading this", positions 2, 3 and 4 are selected to be corrupted. Depending on the chosen type of corruption, "for" is replaced by "`<m>`", "reading" by "cat", which is randomly sampled, while "this" is kept as it is. The loss is applied only at the selected positions, except position 1. The non-corrupted positions are meant to give context for the network to reconstruct the corrupted words, and the network does not get a training signal for these positions to prevent a bias towards copying. Note that the model is supposed to learn to decide whether to trust the input and copy it to the output (position 4) when the input looks correct in the context.

Table 3.1: Illustration of the Cloze task training with different types of input corruption (mask, random replacement, keep).

Original	Thanks	for	reading	this
Operation	-	mask	random replacement	keep
Corrupted	Thanks	`<m>`	*cat*	this
Loss	0	$-\log p_\theta(\text{for}\|\ldots$ `</m>` $\ldots)$	$-\log p_\theta(\text{reading}\|\ldots \textit{cat} \ldots)$	$-\log p_\theta(\text{this}\|\ldots \text{this} \ldots)$

The Cloze task modeling broadly belongs to the class of denoising autoencoders [Vincent & Larochelle$^+$ 08], where the model learns to correct a noisy input while its internal representation is in use. Specifically, it is a particular variant of sequential denoising autoencoder [Hill & Cho$^+$ 16], which processes text input/output. Furthermore, since we put different weights on the positions in training, it is called an *emphasized* denoising autoencoder [Vincent & Larochelle$^+$ 10]. Specifically, the model is trained only on the corrupted positions, i.e. putting zero weight on untouched positions, which puts full emphasis on the corrupted positions. More interpretations and formulations of the Cloze task modeling as denoising autoencoder can be found in [Nix 19].

3.5.3 Inference

In testing, we are interested only in extracting good representations, which we expect the model to have learned via the randomized training for the three tasks (Section 3.5.2). Given a test sentence, we feed it to the trained model without corruption and extract the forwarded states $\mathbf{s}_i^{(l)}$ as the representation vectors. Note again that our purpose in exploiting the Cloze task modeling is not the actual reconstruction of the corrupted input, so we do not pay attention to the output probabilities.

The extracted representations are used in various ways. One can train a classifier for other NLP tasks, e.g. named entity recognition or sentiment analysis, with the representation vectors as its inputs. It is also possible to attach the classifier network directly on top of the representation layers and fine-tune the whole combined network. Either way, the training of a Cloze task model can be regarded as pre-training a part of the parameters for the final NLP task model. We refer the reader to [Devlin & Chang$^+$ 19] and [Peters & Ruder$^+$ 19] for more details of the uses of the representations. In this thesis, we utilize the Cloze task training as a pre-training job for semi-supervised (Section 4.4) and unsupervised NMT (Section 6.4).

3.6 Common Experimental Settings

In this section, we describe experimental settings that are commonly applied throughout this thesis. Unless otherwise noted, all experiments in the following chapters use these settings. Note that all steps described in this section were carried out within Sisyphus [Peter & Beck$^+$ 18] to define their dependencies and manage the workflows.

3.6.1 Data Processing

Both source and target sides of training/development corpora are preprocessed before training. They are first tokenized with the script of the Moses toolkit [Koehn & Hoang$^+$ 07b]. For languages with letter case, frequent casing [Vilar & Stein$^+$ 10] is applied to make the casing of each word consistent over the whole corpus. Various types of whitespaces and punctuation with the same function are also normalized. After that, the granularity of the vocabulary is chosen to be either word or subword. A byte pair encoding (BPE) [Sennrich & Haddow$^+$ 16c] algorithm is used to learn a subword vocabulary from training corpora. The algorithm first splits all words into characters and iteratively merges the most common pairs of characters. The merging is repeated a predefined number of times and a vocabulary is constructed from the resulting tokens. BPE units have proved to be effective in translating a rare word, regarding it not as an out-of-vocabulary (OOV) word but as a sequence of subwords in the vocabulary. Finally, only the sentences with a maximum length of 100 tokens are considered for training. After training, the source side of the test corpora are preprocessed in the same way before decoding.

3.6.2 Model and Training

The NMT models in this thesis follow the Transformer base architecture [Vaswani & Shazeer$^+$ 17] with both the encoder and the decoder having 6 layers. It has hidden states of 512 dimensions, feedforward sublayers of 2048 dimensions, and 8 heads for each attention mechanism (Section 3.3.1). Source and target embedding weights and the output layer linear projection parameters are shared by default.

The models are trained to optimize the cross-entropy with a label smoothing of $\epsilon = 0.1$ (Equation 3.51). Optimization is performed using the Adam algorithm with its default hyperparameter values [Kingma & Ba 15] with a batch size of 4096 tokens. The learning rate is initially set to 0.0001 and decayed by 70% after 3 consecutive checkpoints with no improvement in the development set perplexity, where each checkpoint is defined as 4000 mini-batches. Training is stopped when the development set perplexity does not improve for 8 consecutive checkpoints. During training, dropout is applied with a probability of 0.1 to all sublayers and embeddings.

3.6.3 Decoding and Evaluation

In order to measure the performance of a translation model, we should decode test corpora with the model and evaluate its translation quality in a systematic manner. Sentences in a test corpus must not overlap with the training data; training examples can be memorized by the model during training and it tends to produce nearly perfect translations of them, even if the model has a small and elementary architecture. This does not reflect the real scenario of an MT system where various user inputs are given which are not seen in the training.

Decoding is performed using beam search with a beam size of 5 and a length penalty of 1.0. A target hypothesis generated by a model goes through a series of postprocessing steps. These include merging of all subwords into words, reverting tokenization and normalization [Koehn & Hoang+ 07b], uppercasing the first word of each sentence, and normalizing abbreviations/units.

Translations of test examples can be evaluated by human bilingual experts, but this is expensive and hard to reproduce due to the variability of human judgments. This hinders an efficient and fair comparison of different models, which is crucial for incremental scientific development.

Instead, the MT community mostly uses automatic evaluation metrics to quantify the translation quality objectively. These are mathematical equations with computational algorithms which are designed to be fast, meaningful and reproducible alternatives to human evaluation. These metrics compare the decoded output of an MT system with the *reference* translation, which is considered to be a correct answer to the translation problem. Note that, due to the flexibility of natural language text, there can be multiple answers which are equivalent in semantics and all proper translations. Nevertheless, automatic evaluation metrics are usually computed with a single reference, since preparing multiple references also requires considerable human labor. In this thesis, we use two widely adopted metrics: Bleu [Papineni & Roukos+ 02] and Ter [Snover & Dorr+ 06].

Bilingual Evaluation Understudy

Bilingual evaluation understudy (Bleu) [Papineni & Roukos+ 02] is based on n-gram precisions, i.e. whether n-grams in the hypothesis ($\hat{e}_1^{\hat{I}}$) also exist in the reference (e_1^I):

$$\mathrm{prec}_n(\hat{e}_1^{\hat{I}}, e_1^I) = \frac{\sum\limits_{\hat{e}^{(n)} \in \hat{e}_1^{\hat{I}}} \min\left\{ c(\hat{e}^{(n)}; \hat{e}_1^{\hat{I}}), c(\hat{e}^{(n)}; e_1^I) \right\}}{\sum\limits_{\hat{e}'^{(n)} \in \hat{e}_1^{\hat{I}}} c(\hat{e}'^{(n)}; \hat{e}_1^{\hat{I}})} \tag{3.65}$$

where $\hat{e}^{(n)}$ is a unique n-gram in the hypothesis and $c(\hat{e}^{(n)}; \cdot)$ is the count of how often the n-gram appears in a sentence (hypothesis or reference). Ideally, the count in the hypothesis should match that in the reference. If it is smaller than the count in the reference, the precision score is directly decremented. If it is larger, the count is clipped to the count in the reference by the min operation, eventually getting a lower quantity than the counterpart in the denominator (the count in the hypothesis).

The final formula of Bleu combines all precision scores up to 4-grams by default:

$$\textsc{Bleu}(\hat{e}_1^{\hat{I}}, e_1^I) = \mathrm{bp}(\hat{I}, I) \cdot \prod_{n=1}^{4} \mathrm{prec}_n(\hat{e}_1^{\hat{I}}, e_1^I)^{\frac{1}{n}} \tag{3.66}$$

where bp is a brevity penalty to suppress high scores for very short hypotheses:

$$\mathrm{bp}(\hat{I}, I) = \begin{cases} 1 & \text{if } \hat{I} \geq I \\ \exp(1 - \frac{\hat{I}}{I}) & \text{if } \hat{I} < I \end{cases} \tag{3.67}$$

The scores are then averaged over the full evaluation set to get a corpus-level metric score.

Bleu has been the most popular metric to measure the quality of MT systems and the primary automatic scorer of WMT shared tasks [Koehn & Monz 06, Barrault & Bojar+ 19]. In this thesis, all Bleu scores were computed using SacreBLEU [Post 18], which reproduces exactly the same evaluation pipeline as the WMT tasks.

Translation Edit Rate

Translation edit rate (Ter) [Snover & Dorr+ 06] calculates the number of edit operations (insertion, deletion, substitution, shift) that are needed to transform a hypothesis into the reference translation. This value is divided by the reference length:

$$\textsc{Ter}(\hat{e}_1^{\hat{I}}, e_1^I) = \frac{\text{minimum \#edits from } \hat{e}_1^{\hat{I}} \text{ to } e_1^I}{I} \tag{3.68}$$

In contrast to Bleu, which merely matches word sequences, Ter explicitly takes positional information into account. This makes Ter more interpretable than Bleu with respect to syntactic divergence and allows it to compute more reliable sentence-level scores. In this thesis, all Ter values were computed using TERCOM [Snover & Dorr+ 06], which is case-insensitive by default.

4. Semi-supervised Learning

The first alternative data source to bilingual training corpora is a monolingual corpus, which consists of only a single language (source or target). Such monolingual data is *unlabeled* in the general sense of a classification task, while bilingual data is source text *labeled* with target translations. When using monolingual corpora in MT, we combine labeled and unlabeled examples in training, which is thus called *semi-supervised* learning.

Monolingual data is much easier to acquire for arbitrary languages and genres since there is no need for bilingual sentence alignment. Monolingual corpora do not contain translation examples, but provide additional knowledge about the language structure and fluency for a better translation. They can also have domain-specific text which helps the model adapt to a desired domain.

This chapter covers three methodologies to make use of monolingual data in NMT, in addition to given bilingual data:

1. Integrating an external LM (Section 4.2, modeling and decoding): We revisit the log-linear combination with different levels of normalization. The empirical performance is verified with an extensive hyperparameter search.
2. Generating synthetic bilingual data (Section 4.3, training): We propose new sampling methods for back-translation [Sennrich & Haddow$^+$ 16b] and verify the performance and scalability of all the variants. We also analyze the properties of back-translated data through controlled experiments.
3. Training NMT components with monolingual objectives (Section 4.4, training): We verify the empirical performance of LM pre-training and multi-task learning for NMT [Ramachandran & Liu$^+$ 17] in the Transformer architecture (Section 3.3.1). We propose to use the Cloze task objective (Section 3.5) and perform cross-lingual training in those learning concepts.

At the end, Section 4.5 shows which of the three methods is best in the semi-supervised setting.

4.1 Related Work

Before investigating each methodology, we review the literature up to the time when the work described in this thesis was conducted.

Language Model Integration. Leveraging monolingual data in MT was historically done by including an LM, as this was naturally derived as part of building statistical machine translation models (Section 3.2.1). Also, the traditional phrase-based machine translation systems benefited greatly from an LM, which was a crucial component in the log-linear model combination (Section 3.2.2).

In NMT, [Gulcehre & Firat$^+$ 15] try the log-linear combination between an LM and an NMT model, showing minor improvements over NMT-only baselines. They also combine decoder states and LM states via gating and fine-tune the output layer which depends on the combined states. [Sriram & Jun$^+$ 18] extend this idea by combining the output distributions of both models using dimension-dependent gating, while [Stahlberg & Cross$^+$ 18] replace the gating with multiplication. The last two methods start the NMT training from scratch with a pre-trained LM attached and fixed.

In Section 4.2, we turn the focus back to the classical log-linear combination, which does not require a change of the translation model or training scheme. We revisit its formulation and study its variants and hyperparameters in detail, optimizing the combined performance. We argue that this is the most straightforward and interpretable integration of an LM into NMT, which gives a comparable performance to other integration methods when optimized properly (Section 4.2.3).

Synthesizing Bilingual Data. The concept of synthesizing a bilingual sentence pair from a monolingual sentence was already employed in phrase-based MT [Ueffing 06, Ueffing & Haffari$^+$ 07, Hu & Wang$^+$ 07, Chen & Zhang$^+$ 08, Schwenk 08, Huck & Vilar$^+$ 11]. Such works mostly use source monolingual data to generate the target translations, while additional target monolingual data is used for a target LM. The improvements when adding synthetic bilingual data are marginal in phrase-based MT, considering the amount of generated data.

For NMT, translating target monolingual data to the source side has been shown to improve the performance significantly [Sennrich & Haddow$^+$ 16b]. This is called back-translation; the term had already been introduced in [Koehn 05] but in the context of round-trip translation for evaluating MT performance without bilingual test sets. It was originally shown that a stronger target-to-source model leads to better synthetic data, which in turn improves the source-to-target model that uses such data [Sennrich & Haddow$^+$ 16b, Hoang & Koehn$^+$ 18, Burlot & Yvon 18]. On the other hand, random sampling from the target-to-source model, which is far from an optimal translation, also proves to be effective [Edunov & Ott$^+$ 18, Imamura & Fujita$^+$ 18]. In either case, the use is based mostly on intuitions without a thorough explanation of how it is helpful in training a source-to-target translation model. Section 4.3 offers a unified view of all back-translation variants within the cross-entropy criterion, clarifying the properties of the generation model (the target-to-source model). In the same line of thought, [Cotterell & Kreutzer 18] generalize back-translation in a variational process. This thesis also includes extensive empirical verification of all the variants (Section 4.3.2), including novel sampling methods for back-translation (Section 4.3.1).

Monolingual Training. Training the same model parameters for LM and NMT originates from the multi-task learning principle [Caruana 97]. Multi-tasking for diverse NLP tasks was initiated by [Collobert & Weston 08] and [Collobert & Weston$^+$ 11], while [Luong & Le$^+$ 16] began to combine it with NMT.

The first attempt at pre-training NMT components with LM objectives was done in [Ramachandran & Liu$^+$ 17], which was, however, limited to the RNN-based architectures that were outdated at the time of this work. Section 4.4.3 tests LM pre-training in the Transformer architectures [Vaswani & Shazeer$^+$ 17], including comparison to the multi-task learning. The Transformer LM has been studied as the base of various NLU tasks [Radford & Narasimhan$^+$ 18]. [Devlin & Chang$^+$ 19] propose to predict words given bidirectional context (Section 3.5), which we exploit as another monolingual objective for NMT in this thesis.

[Wada & Iwata 18] show an attempt to train RNN LMs for multiple languages, mainly for word translation tasks. Section 4.4.2 applies this idea in pre-training for NMT with the Transformer architectures. As for pre-training, [Conneau & Lample 19] have been investigating similar extensions concurrently with this thesis.

4.2 Language Model Integration

The initial formulations of SMT [Brown & Cocke+ 90] and phrase-based MT [Zens & Och+ 02] have a separate LM component, which only evaluates the fluency of the output sentence (Section 3.2). The probability provided by an LM can be used to decide between two similar words, where one is more likely to appear in the context of the sentence than the other. In the recent NMT framework, component-wise modeling assumptions are eliminated, making the translation model a more fine-grained LM with additional conditioning on the source sentence. Also, the end-to-end nature of the translation model offers more freedom and expressiveness without the need for explicit word alignments or phrasal segmentations (Section 3.3.1), diminishing the role of an individual LM.

The application of an LM might still be worthwhile in a low-resource scenario, where parallel data is scarce but large monolingual texts are available. Since monolingual texts are more accessible than parallel texts, LMs can be trained on larger amounts of data, either out-of- or in-domain, and learn more intricate properties of the target language.

This section investigates the classical log-linear combination of an LM and a translation model in the NMT context. An LM can be trained independently of an NMT model and later integrated in the decoding process. Compared to LM integration in the training process [Sriram & Jun+ 18, Stahlberg & Cross+ 18], this approach is more efficient, convenient, and interpretable with explicit scaling factors of the two models. In the following, we compare two different normalization methods in the log-linear combination of NMT and LM scores. Section 4.2.3 presents more details on the hyperparameter optimization in decoding and the empirical results.

4.2.1 Sequence-Level Combination

A simple way to combine NMT and LM scores is at the sentence level:

$$Pr(e_0^I|f_1^J) = \frac{p(e_0^I|f_1^J) \cdot p(e_0^I)^\lambda}{\sum\limits_{I', e'^{I'}_0} p(e'^{I'}_0|f_1^J) \cdot p(e'^{I'}_0)^\lambda} \tag{4.1}$$

$$= \frac{p(e_0^I|f_1^J) \cdot p(e_0^I)^\lambda}{\mathrm{Z}(f_1^J)} \tag{4.2}$$

The translation model is combined with the LM probability weighted by $\lambda \in \mathbb{R}$. In order for the model to represent $Pr(e_0^I|f_1^J)$, the combined score must be a valid probability distribution. Accordingly, one normalizes over the sum of all possible events of the model, i.e. all possible target sentences, which results in a normalization factor $\mathrm{Z}(f_1^J)$.

The different modeling leads to the following changes in the decision rule (Equation 3.15):

$$\operatorname*{argmax}_{I, e_1^I} \Big\{ \log Pr(e_0^I|f_1^J) \Big\} = \operatorname*{argmax}_{I, e_1^I} \Big\{ \log p(e_0^I|f_1^J) + \lambda \cdot \log p(e_0^I) - \log \mathrm{Z}(f_1^J) \Big\} \tag{4.3}$$

$$= \operatorname*{argmax}_{I, e_1^I} \Big\{ \frac{1}{I} \sum_{i=1}^{I} \Big[\log p(e_i|e_0^{i-1}, f_1^J) + \lambda \cdot \log p(e_i|e_0^{i-1}) \Big] \Big\} \tag{4.4}$$

with the additive combination in log space. Since $\mathrm{Z}(f_1^J)$ depends only on the source sentence, it can be dropped from the decision. This is comparable to the search criterion of a linear-chain conditional random field [Lafferty & McCallum+ 01]. Compared to the plain NMT, it requires additional operations of the LM query at each position i and a weighted sum of the LM and NMT log-probabilities.

In principle, the LM probabilities should be retrieved for every target word e_i at position i for each hypothesis to be expanded. However, with a count-based LM, it is only possible to query the LM probability of one possible word at a time. This increases the complexity of the additional operations to $O(NV_e)$ for each target position. To speed up the computation, we query the LM probability only for the top-H most likely translation candidates from the NMT model. We call this hyperparameter H *observation histogram size*. The remaining words are given the LM probability of 0. Note that this approximation is not necessary for neural LMs since it obtains a distribution over the whole vocabulary with a single forward step.

For implemention, we can reuse the framework of the NMT search algorithm (Section 3.3.3) and only modify the function that returns the score distributions for each hypothesis. All arithmetic operations are applied element-wise to the translation and LM probabilities.

4.2.2 Position-Level Combination

In the sentence-level combination, we model the conditional probability $Pr(e_0^I|f_1^J)$ directly. Alternatively, this probability can be decomposed into position-level probabilities $Pr(e_i|e_0^{i-1}, f_1^J)$ (Equation 3.14). This word probability can be modeled by a log-linear combination as well:

$$Pr(e_i|e_0^{i-1}, f_1^J) = \frac{p(e_i|e_0^{i-1}, f_1^J) \cdot p(e_i|e_0^{i-1})^\lambda}{\sum\limits_{e \in V_e} p(e|e_0^{i-1}, f_1^J) \cdot p(e|e_0^{i-1})^\lambda} \tag{4.5}$$

$$= \frac{p(e_i|e_0^{i-1}, f_1^J) \cdot p(e_i|e_0^{i-1})^\lambda}{\mathrm{Z}(f_1^J, e_0^{i-1})} \tag{4.6}$$

where the NMT and LM probabilities are combined and renormalized at each position. The decision rule is modified accordingly as follows:

$$\operatorname*{argmax}_{I, e_1^I} \left\{ \log Pr(e_0^I|f_1^J) \right\}$$

$$= \operatorname*{argmax}_{I, e_1^I} \left\{ \frac{1}{I} \sum_{i=1}^{I} \log Pr(e_i|e_0^{i-1}, f_1^J) \right\} \tag{4.7}$$

$$= \operatorname*{argmax}_{I, e_1^I} \left\{ \frac{1}{I} \sum_{i=1}^{I} \Big[\log p(e_i|e_0^{i-1}, f_1^J) + \lambda \cdot \log p(e_i|e_0^{i-1}) - \log \mathrm{Z}(f_1^J, e_0^{i-1}) \Big] \right\} \tag{4.8}$$

The normalization factor $\mathrm{Z}(f_1^J, e_1^{i-1})$ now has an additional dependence on the partial hypothesis e_1^{i-1}, so it cannot be ignored in the decision. For numerical stability, each model score is computed in log space first and then mapped back to probability space for accumulating the normalization factor. At the end, we add all three terms of Equation 4.8 in log space to get the final combined score for position i.

This form is reminiscent of the inference of maximum entropy Markov models [Berger & Pietra$^+$ 96, Ratnaparkhi 96]. Maximum entropy Markov models suffer from the *label bias* problem, i.e. target word expansions do not depend on the full input sequence. This issue is not necessarily present in NMT since the translation model always depends on the whole source sentence. When upscaling the contribution of the LM by increasing λ, however, the source sentence plays a smaller role and the label bias problem arises.

4.2.3 Experiments

We evaluate and analyze the log-linear combination of an LM and an NMT model on two language pairs: WMT 2017 Turkish↔English and WMT 2018 German↔English news translation tasks. The Turkish-English pair is in a typical low-resource condition, where the bilingual data has only around 200k sentence pairs. We prepared monolingual datasets that were around 23 times

Table 4.1: Perplexity of target side LMs (English) on the development set.

LM Type	WMT 2018 de-en		WMT 2017 tr-en	
	Word PPL	Subword PPL	Word PPL	Subword PPL
Count	134.0	75.2	175.1	61.9
Neural	94.0	55.0	126.4	47.7

larger for Turkish and 480 times larger for English. German-English is one of the most researched language pairs and has millions of bilingual sentence pairs. For both German and English, we used 100M monolingual sentences, which is about 17 times larger than the bilingual data. The detailed corpus statistics are given in Section B.1 and B.2.

For count-based LMs, we used the KenLM toolkit [Heafield 11] to train 9-gram models with interpolated modified Kneser-Ney smoothing (Section 3.4.1), while pruning singleton 3-/4-grams and doubleton 5-/9-grams. As neural LMs, we trained 2-layer LSTM models (Section 3.4.2) with 2048 hidden nodes and 512-dimension embeddings. We trained them with SGD with an initial learning rate of 1.0 and a decay of 80% for every two checkpoints of non-improving perplexity on the development set, where each checkpoint amounts to 5000 updates. A batch size of 650 words was used and gradients were not normalized but rescaled when their norm was over 2.0 [Pascanu & Mikolov+ 13]. We stopped the training after 32 consecutive checkpoints with no improvement in the development set perplexity. Table 4.1 shows the perplexity values of the trained LMs. Note that all LMs were trained on the target subword vocabulary of the NMT model.

The neural LMs and the integration of LMs were implemented in Sockeye [Hieber & Domhan+ 17]. The training was run for a maximum of 500k updates in German→English and 300k updates in Turkish→English. The German→English training started from a learning rate of 0.0002, and the Turkish→English training was done with a dropout rate of 0.3.

Hyperparameter Search. During initial experiments, we found that the performance of the LM integration was highly sensitive to the hyperparameters in decoding: beam size, length penalty, LM scaling factor, and observation histogram size (only for count-based models). Each hyperparameter was tuned one at a time using grid search based on BLEU-TER on the development set. In cases where BLEU-TER was the same, we chose the one with the smallest beam size and observation histogram size, which leads to faster decoding.

Table 4.2 lists the hyperparameter values tuned for different configurations of the LM integra-

Table 4.2: Hyperparameter values used for the LM integration experiments.

Task	LM	Combination Level	beam size (N)	LM scaling factor (λ)	length penalty (ω)	observation histogram size (H)
de-en	None	-	25	-	0.7	-
	Count	Sequence	25	0.1	1.0	500
		Position	25	0.15	0.8	25
	Neural	Sequence	15	0.1	0.8	-
		Position	20	0.1	0.7	-
tr-en	None	-	15	-	0.9	-
	Count	Sequence	20	0.25	0.8	20
		Position	25	0.25	0.8	100
	Neural	Sequence	30	0.15	0.5	-
		Position	15	0.1	0.8	-

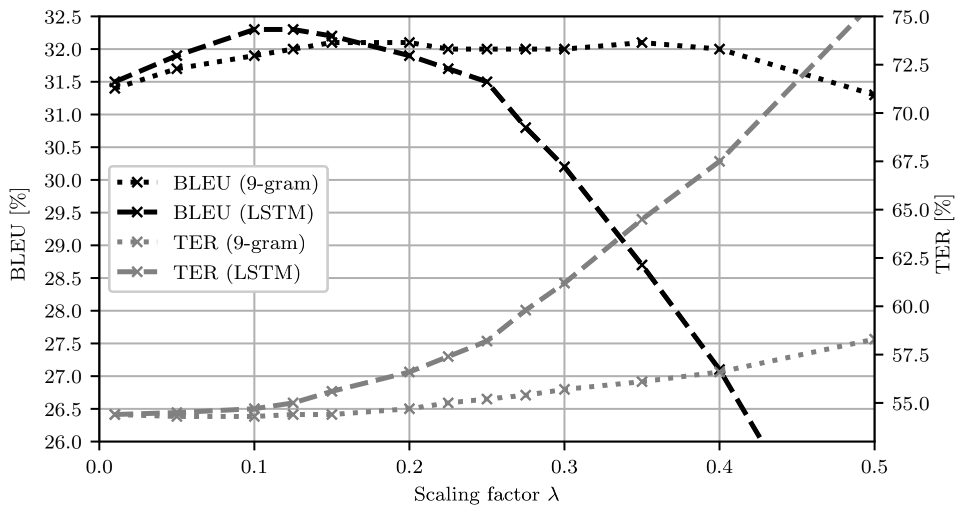


Figure 4.1: Translation performance over the LM scaling factor in a log-linear combination with NMT (WMT 2018 German→English `newstest2015`) [Graça 19].

tion. In general, the best performance could be obtained using a beam size around 20 and a length penalty around 0.8. The observation histogram size has a marginal effect on the performance if it exceeds 10, already covering more than 85% of the probability mass of the NMT model. For the position-level combination, a small observation histogram causes inaccurate normalization factors, yet we do not notice any performance degradation from it.

The most impactful hyperparameter is the scaling factor of the LM, i.e. λ in Equation 4.4 and 4.8. Figure 4.1 shows Bleu and Ter scores for different values of λ. The best choice for λ is between 0.1 and 0.2 and both types of LMs drop in performance for λ larger than 0.4, which indicates that the NMT model should remain as the main contribution to the log-linear combination. The count-based LM is shown to be more robust to the choice of scaling factor than the neural LM. We observed that a neural LM produces more peaked output distributions given long contexts, which thus have a greater effect in the hypothesis search.

Integration Results. Table 4.3 showcases how much a count-based or neural LM improves the standard NMT system with either level of log-linear combination. The LM helps to improve Bleu scores in all scenarios up to +0.9%, as opposed to [Gulcehre & Firat$^+$ 15, Sriram & Jun$^+$ 18, Stahlberg & Cross$^+$ 18], who report little to no improvement with the log-linear integration of an LM. We argue that when the hyperparameters are optimized properly, the log-linear combination is a simple way to achieve significant improvements even without further training. However, the LM-integrated systems show degradation of Ter in many cases. We observed that LMs encourage relatively long hypotheses by adding phrases to increase the fluency, even without an explicit source side context. For instance, spurious commas are added for extra adverb phrases and for a clear distinction between subsentences. This introduces many insertion errors for Ter, which inherently rates short sequences better (Section 3.6.3). This phenomenon occurs more often with a neural LM than with a count-based LM.

Changing the level of the combination does not yield meaningful and consistent improvements across the two tasks. Therefore, considering the efficiency, it is reasonable to use the sequence-level combination which does not involve an extra normalization step for each position (Equation 4.4). An interesting finding is that count-based LMs perform similarly to neural LMs in NMT even though the perplexity values are significantly worse than neural LMs (Table 4.1). This can be attributed to a new type of information from a count-based LM which an NMT model does not

Table 4.3: Translation performances of integrating an LM in NMT. The perplexities of the used LMs can be found in Table 4.1.

		WMT 2017 tr-en				WMT 2018 de-en			
		`newstest2016`		`newstest2017`		`newstest2017`		`newstest2018`	
LM Type	Combination Level	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
None	-	19.0	70.5	18.9	71.1	32.6	53.4	39.2	46.1
Count	Sequence	20.2	69.1	19.5	70.1	**33.5**	**53.8**	**40.1**	**46.4**
	Position	20.3	70.2	19.6	71.0	33.3	53.6	39.7	46.0
Neural	Sequence	20.0	69.6	19.3	70.4	33.5	54.2	39.9	47.0
	Position	**20.3**	**70.1**	**19.6**	**71.0**	33.4	53.8	39.8	46.5

carry. An n-gram model is defined exclusively with discrete statistics of word cooccurrences and computed given restrict context, where an NMT model learns continuous-valued representations with virtually unlimited context. An NMT model may still benefit from the novel information of the count-based LM despite its high perplexities.

4.3 Synthesizing Bilingual Data

Instead of using monolingual data directly to train an external LM, we may also generate bilingual data from it using existing translation models. The generated data can be used to train an MT model along with the original bilingual training corpora. This data augmentation is simple in the sense that it does not modify the model architecture or the learning objective. Using more data is a rule of thumb to improve the performance of a machine learning system, and it should be especially effective in the form that the model is familiar with, i.e. bilingual sentence pairs for an MT model. In addition, we can expect a domain adaptation effect if the monolingual data is close to the test domain.

Such additional data is *synthetic*, compared to the human-generated bilingual data. It has distinct properties in linguistic statistics, which are analyzed systematically in Section 4.3.2. In the same section, we also verify that the synthetic translation might not necessarily be the optimal hypothesis of an MT model.

We first describe the general process of the synthesis in the case of generating source side given target monolingual data, but the opposite case is analogous. Given a target monolingual corpus $\mathcal{C}_e$, synthetic bilingual data is generated using a target-to-source translation model p_{t2s} which is trained with real bilingual training data $\mathcal{C}_{f,e}$. In low-resource scenarios (Section 1.2), p_{t2s} also tends to be of low quality and biased towards the small bilingual data. To mitigate the low quality of p_{t2s}, we may introduce an additional control on its output; this leads to broader synthetic data generation schemes, including sampling from the target-to-source model. When training the main translation model p_θ (source-to-target) with such synthesized data, we also use the same cross-entropy criterion (Section 3.3.2) as supervised learning.

4.3.1 Generation Strategies

In this section, we compare various strategies for generating a synthetic source sentence from a given p_{t2s}. A generation strategy potentially imposes limitations on the probability distribution p_{t2s}, from which we draw source sentence samples. In practice, changing generation strategy is choosing another inference method of p_{t2s}. This does not involve retraining of a model so it is less arduous than modifying the target-to-source model p_{t2s}. In the following, we cover the first proposed strategy of [Sennrich & Haddow$^+$ 16b], the sampling strategy of [Edunov & Ott$^+$ 18], and the original methods of this thesis.

Decoding

The most straightforward generation strategy is normal decoding (Section 3.3.3). It returns the highest scoring sentence according to the search criterion, which is often approximated by beam search and widely known as back-translation [Sennrich & Haddow$^+$ 16b]. Decoding is a deterministic process and produces practically the best translation that can be generated by model p_{t2s}. This translation usually takes a large portion of the probability mass of p_{t2s} and is considered as a good representative among all possible f_1^J.

However, decoding with beam search tends to have a high bias towards the more common variant of translation [Ott & Auli$^+$ 18]. The bias also exists at the sub-sentence level for words or phrases which are relatively independent of the context of the remaining sentence. For example, the German word "Hund" might be translated to both "dog" and "hound" in English, which are both valid translations. Beam search will choose the popular word "dog" most of the time and the resulting synthetic data will have a skewed distribution, eliminating many natural variations in translation. Decoding-generated synthetic data also has a bias in reordering, as we show in Section 4.3.2. We argue that these biases hamper correct approximation of the bilingual sentence pair space. Also, the skewness caused by the biases is even more intensified if a sentence is present in the target monolingual data multiple times in the same or similar forms.

Unrestricted Sampling

To counteract the deterministic bias of beam search, one might employ probabilistic sampling from the target-to-source model [Edunov & Ott$^+$ 18], where the synthesis process coincides with the model generation probability. This is implemented via sampling a word sequentially over j from the conditional probability $p_{\text{t2s}}(f_j|f_1^{j-1}, e_0^I)$ until the sentence-end symbol is reached. Each word is sampled from the model weighted by the word probabilities, i.e. via a multinomial process.

Probabilistic sampling results in different translations of a sentence, achieving more variability in their syntax and semantics. It does not limit the search to the maximum probability output and can have a good coverage of the whole space of source sentences. Nevertheless, when an erroneous word with a very low probability is sampled, the model begins to condition the following word probabilities on that word and consequently generates non-parallel translations. Such non-parallel sentence pairs influence the main model training more severely when the number of source samples per target sentence is small.

Interestingly, the erroneous words are actually promoted by the current state-of-the-art training for NMT, which assigns more probability mass to sentences that should have obtained a smaller probability [Ott & Auli$^+$ 18]. This probability smearing phenomenon is made even worse by label smoothing (Section 3.3.2), which teaches the model to partially fit a uniform word distribution. Figure 4.2 provides the average cumulative probabilities of the top-N words for NMT models, trained with and without label smoothing. We observe that label smoothing causes a reallocation of roughly 7% probability mass to all except the top-100 words. This reallocation is not problematic in beam search which only looks at the top-scoring candidates. However, in sampling-based generation, it leads to more words with low quality in the sampled sentences that stray away from the distribution of natural data [Edunov & Ott$^+$ 18]. Note that a model that fits well to the true distribution must not only assign a high probability to high-quality translations, but also assign a low probability to low-quality translations.

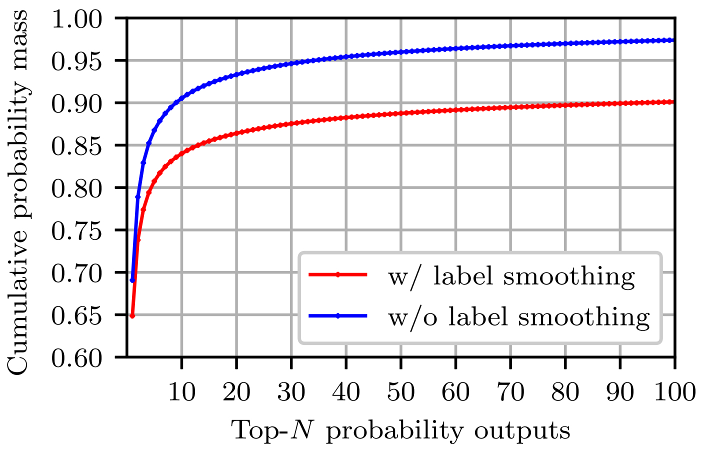


Figure 4.2: Cumulative probabilities of the top-N word candidates with and without label smoothing (WMT 2018 English→German `newstest2015`).

τ-restricted Sampling

To prevent the sampling from generating nonsense translations, it is necessary to restrict the sampling space to high-probability sentences. This provides a middle ground between unrestricted sampling and beam search with respect to the diversity of generated sentences. [Edunov & Ott$^+$ 18] consider sampling from a predefined number of top-scoring words, which has, however, no guarantee that the top candidates are all reasonable translations. Alternatively, we propose to restrict the sampling space to a set of words with a minimum probability $\tau < 0.5$:

$$\mathcal{V}_j^\tau = \left\{ f \mid p_{\text{t2s}}(f|f_1^{j-1}, e_0^I) \geq \tau \right\} \tag{4.9}$$

for each position j in the sequential generation of source words. The target-to-source model probabilities are renormalized over those words in $\mathcal{V}_j^\tau$. This is implemented in practice by applying the softmax function to the logit entries for the words in $\mathcal{V}_j^\tau$ where all the other entries are dropped. Depending on the word distribution for each position j, $\mathcal{V}_j^\tau$ may have various sizes. When $\mathcal{V}_j^\tau$ is empty, the model has no confident predictions and it is dangerous to consider multiple candidates; we just select the best candidate and continue to the next position. Note that the method gets closer to greedy search with a larger τ and closer to unrestricted sampling with a smaller τ.

N-best List Sampling

Another way to sample only from probable hypotheses is to generate a list of N-best hypotheses $\mathcal{H}_{p_{\text{t2s}}}^N$ and sample from them. The hypotheses are assigned a probability based on softmax-renormalized scores in decoding. The higher the probability of a sentence, the higher the weight assigned to that sentence in the sampling process. In contrast to the τ-restricted sampling, it constrains the search space by the sequence-level score. A drawback of an N-best list is that it might contain only small variations of the best output, e.g. replacing one word or dropping an article.

4.3.2 Experiments

The presented synthetic data generation concepts were implemented in the Sockeye toolkit [Hieber & Domhan$^+$ 17] and evaluated on the same datasets as Section 4.2.3.

Properties of Synthetic Data

Firstly, we analyze how the synthetic examples are different from the original (human-written) bilingual sentence pairs. For this comparison, we divide the original bilingual data of WMT 2018 German→English into two portions, only one of which (randomly selected 1M sentence pairs) is considered in the baseline supervised training and the target-to-source model training. The other (around 5M sentence pairs) is for the synthetic data generation, where its target side is back-translated using the strategies of Section 4.3.1. This setup makes the domain consistent between the bilingual data for supervised training and the target monolingual data for back-translation, where we are able to analyze the synthesized examples more distinctly without a domain adaptation effect. It also provides an upper bound on the performance of a semi-supervised system by training a supervised system on the full original bilingual corpus [Fadaee & Monz 18]. For the semi-supervised systems, we trained the source-to-target models from scratch on the baseline bilingual and synthetic data concatenated.

Lexical Distribution. We first examine the lexical choices of synthetic source sentences as translations of target sentences. For an approximated overview of the lexical translations, we trained a target-to-source IBM-1 model [Brown & Della Pietra$^+$ 93] on the synthetic and natural bilingual data using GIZA++ [Och & Ney 03]. Table 4.4 reports the entropy of the lexicon probabilities and the average number of source words having a nonzero translation probability from a target word. Natural data tends to have a large number of translation candidates per target word and also a high lexicon entropy. This can be explained by the variability of human-generated data that often includes spurious words or synonyms of translations. Synthetic data generated by unrestricted sampling obtains even higher values than the natural data for both statistics. Note that the word-by-word sampling may lead to cases where an improbable word is sampled and the model starts to lose context in the subsequent positions, which produces nonparallel and gibberish sentence pairs. In contrast, synthetic data generated by beam search has a much lower entropy and also a smaller number of nonzero lexicon probabilities than the real data, which indicates much less variability in the translations. The restricted sampling stays in the middle of the two extremes and shows closer statistics to the real data.

Table 4.4: Statistics of the IBM-1 target-to-source lexicon probability trained on synthetic and real bilingual corpora (WMT 2018 German→English). Restricted sampling is performed with $\tau = 10\%$.

Type	Generation Strategy	Entropy	Average Nonzero Entries
Real	-	2.91	129
Synthetic	Beam search	2.59	98
	Unrestricted sampling	3.13	196
	Restricted sampling	**2.66**	**104**

Degree of Reordering. To analyze reorderings in synthetic bilingual data, we learned its word alignments using `fast_align` [Dyer & Chahuneau$^+$ 13] and represented the data as joint translation and reordering (JTR) sequences [Guta & Alkhouli$^+$ 15]. These sequences consist of special tokens which stand for either translation or reordering action, which inform us explicitly of nonmonotonic translations, unaligned words, and multiply aligned words. Table 4.5 shows the distributions of the JTR tokens in the real/synthetic data. The number of unaligned words and reorderings in the real data stems from free translations by human and noisy sentence alignments. The synthetic generation methods tend to produce fewer unaligned words and reorderings, while having more one-to-one translations. Again, restricted sampling finds itself in between beam search

Table 4.5: Percentages of JTR token classes in the synthetic and real bilingual corpora (WMT 2018 German→English).

		Proportion [%]				
Type	Generation Strategy	Delete	Insert	Multi-align	Reorder	Translate
Real		11.1	10.2	8.7	22.4	47.7
Synthetic	Beam Search	7.6	7.7	8.8	20.1	55.8
	Restricted Sampling	**8.1**	**8.6**	8.8	**20.7**	**53.9**

and the real data.

Example Sentence Pairs. To have a more tangible view of the outputs of back-translation, we select three example sentence pairs from the real/synthetic corpora in Table 4.6. The first sentence shows a case where the target sentence lacks a lot of information present in the original source. Both beam search and restricted sampling provide appropriate translations and sampling produces a gibberish output. The second sample presents an example where the word "response" was translated to its most common translation "Antwort" by beam search. The preference for frequent words by beam search is also reflected in the low entropy (Table 4.4). The restricted sampling chooses a proper alternative "Reaktion", which shows its ability to give more variety to translations. In the third example, unrestricted sampling has trouble in ending the output, attaching garbage words which are not related at all to the input sentence but merely extend the output length. Again, restricted sampling chooses a reasonable translation of "Saga" ("Sage"), which is indeed closer to the human translation, while beam search prefers a more commonly used word ("Geschichte").

Table 4.6: Example sentence pairs from real and synthetic bilingual corpora (WMT 2018 German→English).

Target	-	let 's take education as an example :
Source	Real	lassen Sie mich dieses Konzept anhand des Beispiels Bildungspolitik verdeutlichen :
	Beam search	nehmen wir die Bildung als Beispiel :
	Unrestricted sampling	ein Beispiel : CO2 "
	Restricted sampling	nehmen wir die Bildung als Beispiel :

Target	-	so that is the first response .
Source	Real	dies ist nun die erste Stellungnahme .
	Beam search	das ist also die erste Antwort .
	Unrestricted sampling	lagerhaltige Marken können zoomen (cont.)
	Restricted sampling	dies ist also die erste Reaktion .

Target	-	it is seen as a long saga full of surprises .
Source	Real	es wird als eine lange Saga voller Überraschungen angesehen .
	Beam search	es wird als eine lange Geschichte voller Überraschungen angesehen .
	Unrestricted sampling	es wird als eine lange Saga voller Überraschungen angesehen . injury , Skepsis , Feuer) , Duschen verursachter Körper , Paläste , Golfen , Flur und Muffen , Diesel- Total Babylon , der durchs Wasser und Wasserkraft fliet .
	Restricted sampling	es wird als lange Sage voller Überraschungen angesehen .

Difficulty of Training. In Figure 4.3, we check how the NMT training progresses when learning on the synthetic or real data. The data synthesized with beam search is easier for a model to fit, since it includes less ambiguity in lexical choices and more monotonic translations. On the other hand, the real data yields a much higher training perplexity because of its linguistic variety. Free sampling from a target-to-source model shows a similar perplexity curve as that of the real data. By restricting the sampling, the training becomes a bit harder than with the beam-search synthetic data and closer to the variety of the real data.

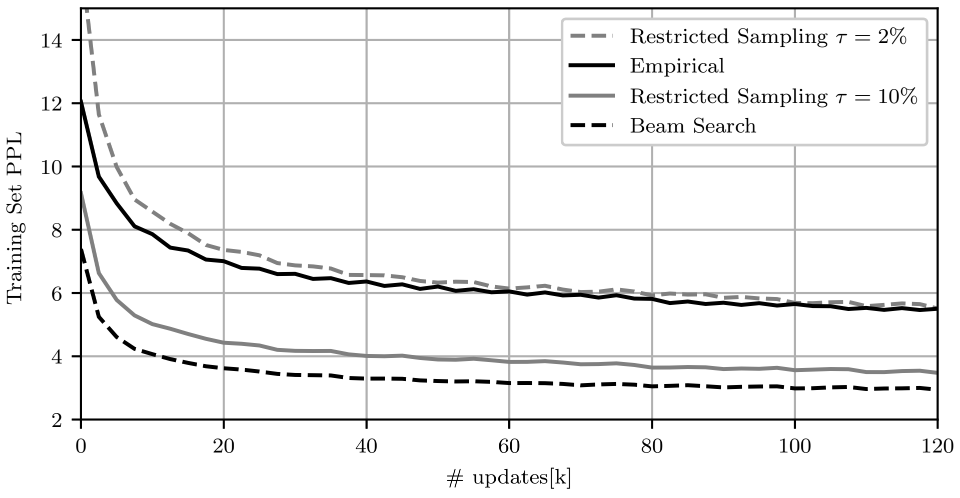


Figure 4.3: Training perplexity progress for augmenting the training corpus with real or various synthetic data [Graça 19].

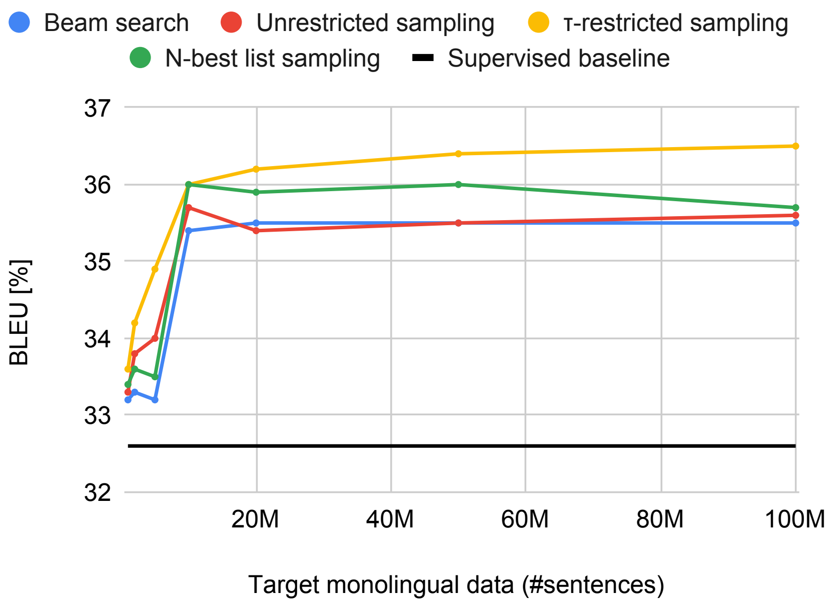
Table 4.7 shows the performance after the training with the additional real/synthetic data. As expected, adding the real bilingual data gives the best performance, which cannot be achieved by using synthetic data. Among the synthetic generation strategies, unrestricted sampling shows the worst performance in both test sets due to many nonparallel sentences, as also shown in Table 4.6. Restricted sampling or N-best list sampling reaches a similar performance to beam search. This shows that sampling from the target-to-source model can be, with proper reduction of the sampling space, a comparable alternative to beam search. Considering Figure 4.3, training fitness does not correlate with the final translation performance.

Table 4.7: Translation performance of augmenting the training data with real or synthetic bilingual corpora (WMT 2018 German→English).

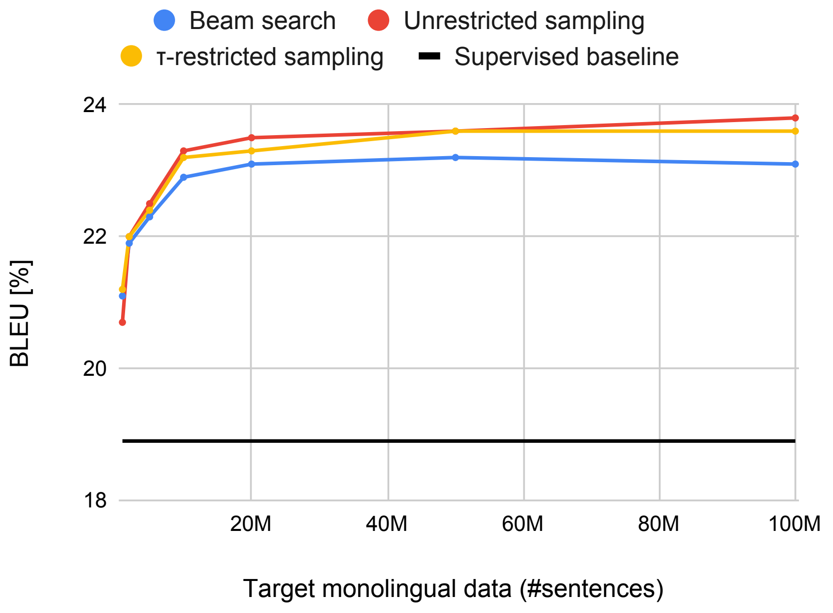
		`newstest2017`		`newstest2018`	
System	Additional Data	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
Supervised	-	28.7	58.4	34.4	51.3
	Real	33.3	53.6	40.3	45.5
Semi-supervised	Beam search	31.9	55.5	**40.1**	**45.7**
	Unrestricted sampling	31.0	55.9	37.9	47.8
	Restricted sampling ($\tau = 10\%$)	**32.1**	**55.0**	39.8	45.8
	N-best list sampling ($N = 50$)	31.9	55.4	39.8	45.9

Comparison of Generation Strategies at Scale

Previously, we have investigated the distinguishing properties of synthetic and natural bilingual corpora by artificially splitting the original training data into two parts. Here, we evaluate different synthetic data generation strategies in realistic scenarios: the original WMT tasks with extra in-domain target monolingual data for the data synthesis (Section B.1 and B.2). The synthesized



(a) German→English `newstest2017`



(b) Turkish→English `newstest2017`

Figure 4.4: Comparison of synthetic data generation strategies.

data were appended to the original bilingual data, where the original data was oversampled so that the ratio of the original/synthetic data is roughly 1:1. When using such combined data (semi-supervised), we continued the training from the parameters of the supervised baseline to speed up the convergence on larger scales. Empirically we found that there is no significant difference in the performance between training from scratch and fine-tuning the baseline model.

Figure 4.4 compares the synthetic data generation strategies of Section 4.3.1 with increasing the target monolingual data size. In both translation tasks, all generation strategies are effective in synthesizing data that improves the translation quality of the main model. The improvement saturates with more than 50M sentence pairs. For efficiency, nondeterministic sampling methods (unrestricted, τ-restricted) scale well with larger data, while beam search and N-best list sampling, which involve best-probing decoding steps, make for a serious increase in the synthesis time.

Among the generation strategies, beam search is the worst in both tasks. We attribute this to its tendency to prefer frequent words and monotonic translations (Section 4.3.2). As opposed to Table 4.7, sampling without restriction is not inferior to beam search and even clearly better (up to +0.7% Bleu) in the Turkish→English task. In the artificial setup of Table 4.7, the target text for the synthetic data is from mixed, out-of-domain bilingual training data, while the synthetic data in Figure 4.4 is generated from the in-domain target language text. We observe that the ill-formed source sentences generated by the unrestricted sampling are indeed harmful if it is outside the test domain. On the other hand, if the target domain of the synthetic data matches that of the test scenario, they are still effective in terms of the domain adaptation. We argue that they encourage a stronger adaptation than the beam search generated data in a sense, which forces the model to choose in-domain expressions even from noisy inputs.

τ-restricted sampling performs comparably to or better than the other synthetic data generation methods in all cases. Note that it has no degradation in performance even with out-of-domain monolingual data (Table 4.7). It introduces only reasonable variations on the target-to-source model output to have a good fit to the training criterion, which appears to be the strongest and the most consistent approach. The performance of N-best list sampling is placed between τ-restricted sampling and beam search.

Language Model Integration in the Synthesis

We conducted extra experiments with a source side LM integrated into the synthetic data generation process. The integration is done at the sequence level (Section 4.2.1) for decoding and N-best list sampling strategies, while the position-level combination (Section 4.2.2) is used in τ-restricted sampling since a word is to be sampled from a normalized distribution for each position. The additional LM changes the generating distribution p_{t2s} directly, instead of choosing a different generation strategy. In Section 4.2, we saw that including an LM in the hypothesis search can

Table 4.8: Effect of a source side LM in generating synthetic bilingual data (WMT 2018 German→English). All LMs are built with LSTM neural networks.

Synthetic Data		`newstest2017`		`newstest2018`	
Generation Strategy	Source LM	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
Beam search	No	35.7	51.0	43.6	42.0
	Yes	35.8	50.9	43.6	42.0
τ-restricted sampling	No	35.9	50.6	43.0	42.4
	Yes	36.0	50.4	43.1	42.3
N-best list sampling	No	35.8	50.8	43.4	42.2
	Yes	36.0	50.5	43.1	42.2

improve the translation quality. When it is included in the generation of source sentences, we expect synthetic data with better parallelism and/or closer relation to the test domain. However, the LM-assisted synthetic data does not produce a meaningful improvement in the performance of the main source-to-target model (Table 4.8). This confirms again that, for synthetic bilingual data, the translation quality is not as important as the diversity and coverage of the source sentence space, as was shown by the effectiveness of sampling-based methods in Figure 4.4.

4.4 Monolingual Training

Modern sequence-to-sequence NMT models (Section 3.3) have a decoder which resembles the behavior of an LM. It estimates a probability distribution over a vocabulary, conditioned on the history of previous words. Many of the parameters of an NMT decoder have the same roles as they do in a neural LM except for the encoder-decoder attentions (Section 3.4.2). This parameter overlap also holds for the encoder of an NMT model; it may share lots of parameters with a Cloze task model (Section 3.5.1). Based on this fact, this section investigates training of NMT parameters for language modeling or Cloze task modeling, i.e. monolingual training objectives.

By training a single network for multiple tasks, the knowledge of one task is related to another task via continuous transformations within the representation space without an explicit mapping of discrete entities [Thrun & Pratt 12]. It also offers better regularization which is motivated by the task similarity, as opposed to heuristics like early stopping, which merely avoid overfitting [Caruana 97]. This is inspired by human learning, as it has been shown that humans are able to learn in a continual fashion [Cichon & Gan 15] across multiple tasks, with each subtask benefiting from the others. In this section, we implement the multi-tasking methodology between bilingual (translation) and monolingual (language modeling, Cloze task modeling) tasks. Among the learning tasks, the transferor task from which we borrow the knowledge is called the *parent*; the transferee task, which reuses the parent model and exploits its knowledge, is called the *child*. The child is the task that we aim to do and optimize for in the end, whereas the parent task is addressed mainly to learn transferrable knowledge for children tasks.

Monolingual training is unsupervised in the context of MT, since it does not require any labels in the form of an aligned sentence translation. It is thus comparatively cheap to obtain training data, which leads to a much larger training corpus compared to the bilingual data. This can be helpful especially for low-resource translation, where the translation objective alone does not generalize well by itself. In the following, we first revisit the monolingual objectives of Section 3.4 and Section 3.5 from the NMT's viewpoint. Next, we present different strategies for using a monolingual objective in NMT training in terms of data scheduling and cross-linguality.

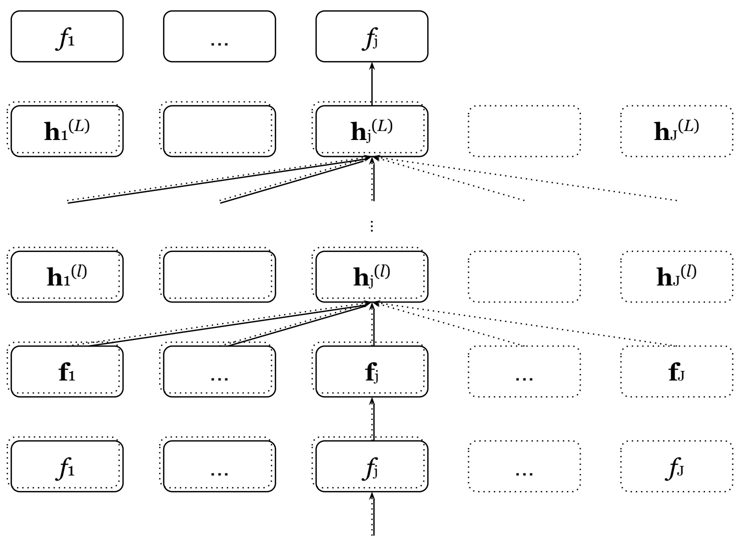
4.4.1 Training Objectives

For each objective, we show which of its modeling parameters overlaps with a Transformer NMT model (Section 3.3.1) and what one can expect by training for it.

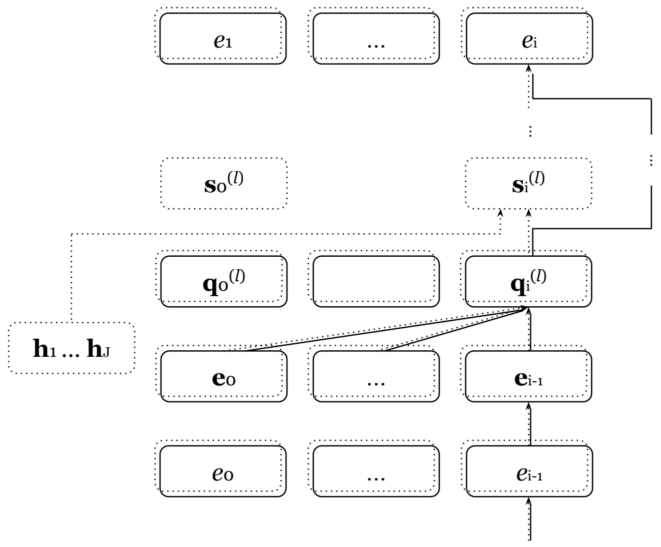
Language Modeling

The first training objective we consider is the traditional left-to-right language modeling (Section 3.4.2). In this setting, both encoder and decoder parameters are trained as part of a Transformer LM on the monolingual data for source and target side respectively. For a smooth transfer of parameters, it requires a careful choice of parameters for each of the LMs.

The Transformer decoder is already practically a left-to-right target LM by design, except for the encoder-decoder attentions. It is thus relatively simple to train the decoder parameters on monolingual data by just ignoring the encoder-decoder attention (Figure 4.5b). The objective



(a) Encoder



(b) Decoder

Figure 4.5: Illustration of NMT model parameters (dotted) and the left-to-right language modeling parameters (solid).

function for training of the decoder components is therefore the same as Equation 3.56, also using the NMT output layer. The decoder state $\mathbf{s}_i^{(l)}$ is computed only from the target self-attention, skipping the encoder-decoder attention.

Training the Transformer encoder as a left-to-right LM is, however, less straightforward in comparison. The self-attention span of the encoder is unrestricted, i.e. it can access every position $1, \dots, J$ from the previous layer, which would ignore the causal relationship of the positions from left to right. It is therefore essential for the LM training to mask out the right-side context for every self-attention layer, in the same way as is done for the decoder. This does not change the number of parameters that are used; however, it causes a mismatch of the input to each layer between the LM training and the NMT training. The encoder for NMT does not have an output layer, but an LM does. Thus a linear layer with subsequent softmax activation has to be added above the topmost encoder layer in the LM training. This will not add any new parameters to the model if the linear layer is shared with the input embeddings (Figure 4.5a). This has the consequence that, among the NMT model parameters, only the encoder-decoder attention is initialized randomly and trained from scratch.

Through the LM training, an NMT decoder is expected to learn more about the fluency of the target language from additional monolingual data. An encoder does not have to predict source words as an LM does, but it still learns useful representations and source sentence structures from the additional task and data.

Cloze Task Modeling

Training for the left-to-right language modeling is natural for the decoder components, but it suffers from a severe mismatch between the LM and NMT training on the encoder side. On one hand, a Transformer LM disregards the right-sided context for every position in the self-attention, and on the other hand, the NMT encoder uses the full self-attention context everywhere. This discrepancy could force the model to disregard LM-trained parameters, since they are not optimized for bidirectional context. Even worse, it could lead to a model that is biased towards left-sided context.

One approach to training the encoder with unmasked self-attention is to use it as a Cloze task model (Section 3.5). The architecture of a Cloze task model is equivalent to that of a Transformer encoder except that it has an output layer. In contrast to a left-to-right LM, the context in the self-attention layers remains unconstrained, which provides more expressive representations. Figure 4.6a shows the overlap of parameters between an NMT encoder and a Cloze task model. In contrast, a fully bidirectional Cloze task model is overparametrized for the NMT decoder, which constrains the target self-attention to the previous context for the left-to-right generation (Figure 4.6b).

The Cloze task modeling is optimized for learning more advanced and universal representations than language modeling, which is helpful for both the encoder and decoder of an NMT model. Compared to language modeling, it does not teach the NMT model to decode target tokens from left to right in the way the translation output is generated.

4.4.2 Training Strategies

In this section, we explain when and how to exploit the monolingual objectives along with the translation model training.

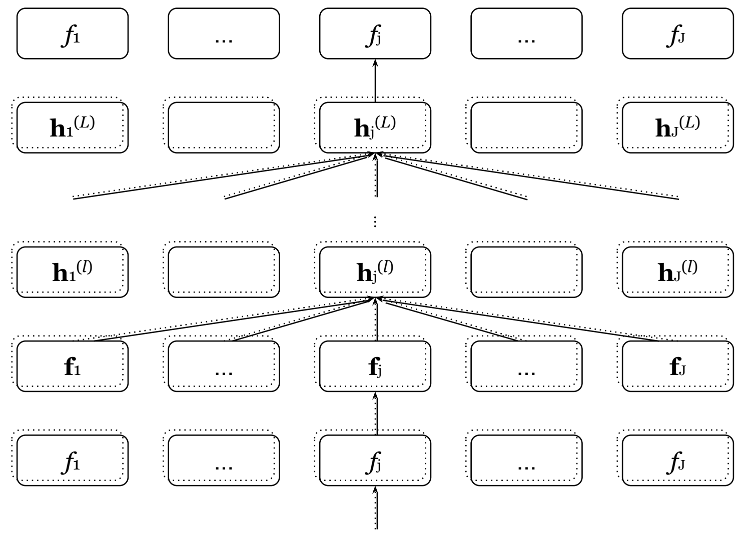
Multi-Task Learning

The first strategy is to learn tasks in parallel with shared parameters. This multi-task learning can be done with the combined loss of a source monolingual objective (L_f), a target monolingual objective (L_e), and the usual translation objective (Equation 3.44, denoted as $\mathrm{L}_{\mathrm{f,e}}$):

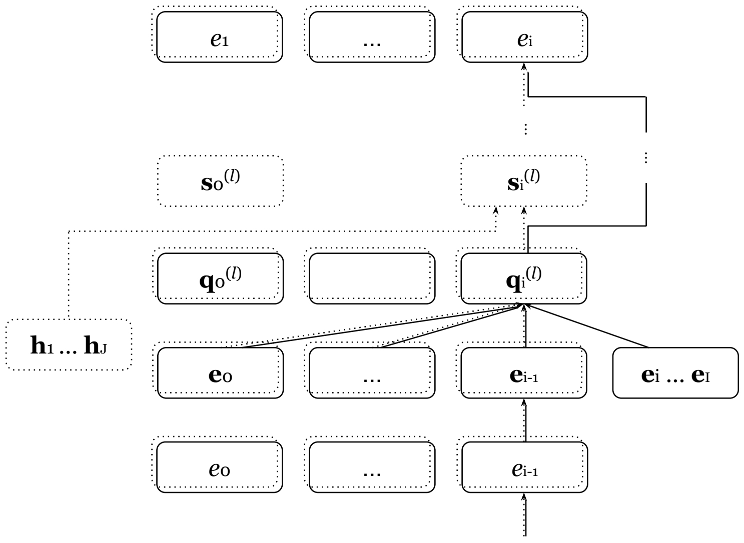
$$\mathrm{L}(\theta) = \lambda_\mathrm{f} \cdot \mathrm{L}_\mathrm{f}(\theta_\mathrm{f}; \mathcal{C}_f) + \lambda_\mathrm{e} \cdot \mathrm{L}(\theta_\mathrm{e}; \mathcal{C}_e) + \lambda_{\mathrm{f,e}} \cdot \mathrm{L}_{\mathrm{f,e}}(\theta; \mathcal{C}_{f,e}) \tag{4.10}$$

with $\lambda_\mathrm{f}, \lambda_\mathrm{e}, \lambda_{\mathrm{f,e}} \in \mathbb{R}^+$ as a weight for each task. θ_f and θ_e denote encoder and decoder parameters respectively, where $\theta_{\mathrm{f-e}}$ indicates encode-decoder attention parameters. The total parameter set is defined as $\theta = \{\theta_\mathrm{f}, \theta_\mathrm{e}, \theta_{\mathrm{f-e}}\}$.

There are three separate forward passes on the respective parameters/data and one combined backward pass with the weighted sum of all three losses. Alternatively, one can also switch to each individual objective for each mini-batch. By optimizing monolingual and bilingual objectives together, we expect the model parameters to be less biased to the bilingual data, which is not



(a) Encoder



(b) Decoder

Figure 4.6: Illustration of NMT model parameters (dotted) and the Cloze task modeling parameters (solid).

representative of the translation task if low-resourced. On the other hand, there is a risk of being overoptimized to the monolingual objectives, which can, however, be controlled with task weight hyperparameters. Note that there is no distinction among parent and child tasks in the multi-task learning.

Sequential Transfer

Another approach is to learn tasks in a sequential order:

1. Train encoder parameters of the NMT model with a source monolingual objective $\mathrm{L}_{\mathrm{f}}(\theta_{\mathrm{f}}; \mathcal{C}_f)$ and decoder parameters with a target monolingual objective $\mathrm{L}_{\mathrm{e}}(\theta_{\mathrm{e}}; \mathcal{C}_e)$.
2. Train the whole NMT model with the translation objective $\mathrm{L}_{\mathrm{f,e}}(\theta; \mathcal{C}_{f,e})$ $(\theta_{\mathrm{f}} \cup \theta_{\mathrm{e}} \subseteq \theta)$.

The goal is now to transfer knowledge from the monolingual tasks (parent) to the translation task (child). A clear advantage of sequential transfer is that the model is optimized for the desired task (translation) at the end.

A potential problem is *catastrophic forgetting*: the learning system forgets the knowledge of the first task while learning the second one [McCloskey & Cohen 89, Ratcliff 90]. In sequential transfer, this means that the monolingual loss increases as the NMT training progresses. This may happen since the NMT training overwrites its knowledge on the same model parameters that are used in the monolingual training. Even if the language modeling or Cloze task performance is not our goal, the benefit of the transfer may be diminished if the NMT training unlearns the knowledge that was obtained in the monolingual objectives. A possible solution is to protect the parameters that are important for the monolingual tasks from drastic changes during the translation training [Kirkpatrick & Pascanu$^+$ 17], or just to apply dropout [Goodfellow & Mirza$^+$ 14]. In this thesis, we adopt a more straightforward approach: continue training on the monolingual objective jointly with the NMT training to keep the language modeling or Cloze task performance high even after the transfer [Ramachandran & Liu$^+$ 17]. This is basically performing multi-task learning (Section 4.4.2) after the monolingual pre-training, where we use the same monolingual data in both learning stages.

Cross-lingual Training

Although we consider monolingual auxiliary objectives, our main task is still translation which handles the relation between source and target languages. The translation task is performed by a close connection between the encoder and the decoder, but the auxiliary pre-training is executed independently for the encoder and the decoder. The two pre-trained components are not coordinated at first and require a further adaptation which learns a mapping from the encoder representation space to the decoder representation space.

In order to increase the correlation between the representation spaces of the two languages, we may also pre-train a single set of parameters $\theta_\mathrm{f} = \theta_\mathrm{e}$ with the union of source and target monolingual data $\mathcal{C}_\mathrm{f} \cup \mathcal{C}_\mathrm{e}$. The combined data is shuffled altogether again before the monolingual pre-training. The trained parameters can be used to initialize both encoder and decoder and optimized separately on the translation task thereafter. This cross-lingual training can be performed for any objective of Section 4.4.1, provided there is a joint vocabulary over the source and target languages.

4.4.3 Experiments

The monolingual training of an NMT model is evaluated on two news translation tasks: WMT 2016 Romanian→English (Section B.3) and WMT 2017 Turkish→English (Section B.1). In both tasks, we used the News Crawl 2014-2015 portion for the English monolingual data (around 55M sentences). The Romanian→English task has bilingual training data of around 600k sentence pairs, which exceeds our definition of a low-resource task (Section 1.2) but yet cannot be called data-rich. For this task, we set the dropout rate to 0.2 and the batch size to 7200 tokens. The Turkish→English experiments used a dropout rate of 0.3 and a batch size of 4096 tokens. For both tasks, a checkpoint was defined as 400k training examples for monolingual pre-training and 200k training examples for translation training or multi-task learning. The multi-task learning used equal weights for all training objectives, i.e. $\lambda_\mathrm{f} = \lambda_\mathrm{e} = \lambda_\mathrm{f,e} = 1.0$ (Equation 4.10). We conducted the experiments in the RETURNN framework [Doetsch & Zeyer$^+$ 17].

Multi-Task Learning vs. Sequential Transfer. First of all, we compare sequential transfer and multi-task learning for the two monolingual objectives in Table 4.9. The monolingual training was applied equally to both the encoder and the decoder. Except in the Turkish→English

Table 4.9: Comparison of sequential transfer and multi-task learning with monolingual training for NMT. The encoder and the decoder are trained with monolingual data of source and target side, respectively.

(a) WMT 2016 Romanian→English task

Data Size (#sents)			Monolingual		`newsdev2016`		`newstest2016`	
					Bleu	Ter	Bleu	Ter
ro-en	ro	en	Objective	Training Strategy	[%]	[%]	[%]	[%]
612k	-	-	-	-	35.5	51.3	33.7	53.7
612k	2.3M	55M	LM	Sequential	**36.6**	**50.1**	34.7	52.8
				Multi-task	35.5	51.5	33.4	53.9
				Sequential + Multi-task	35.9	51.0	33.7	53.7
			Cloze	Sequential	36.3	50.4	**34.7**	**52.6**
				Multi-task	34.9	51.9	33.1	54.2
				Sequential + Multi-task	35.8	50.8	34.0	53.5

(b) WMT 2017 Turkish→English task

Data Size (#sents)			Monolingual		`newstest2016`		`newstest2017`	
					Bleu	Ter	Bleu	Ter
tr-en	tr	en	Objective	Training Strategy	[%]	[%]	[%]	[%]
207k	-	-	-	-	19.0	70.5	18.9	71.1
207k	4.8M	55M	LM	Sequential	19.6	70.1	19.2	70.3
				Multi-task	**20.2**	**69.1**	19.9	69.8
				Sequential + Multi-task	19.7	69.4	19.4	69.8
			Cloze	Sequential	20.0	68.5	**19.8**	**69.2**
				Multi-task	19.7	69.1	19.5	69.7
				Sequential + Multi-task	19.5	69.6	19.3	69.8

task with a language modeling objective, sequential transfer alone ("Sequential") shows a larger improvement than multi-task learning from scratch ("Multi-task") or from monolingually pre-trained parameters ("Sequential + Multi-task"). Multi-task learning improves over the baseline in the Turkish→English task, but in general, optimizing solely for the end task (translation) at the end proves to be better when we evaluate the model only on that task. For each training strategy, it is difficult to draw a conclusion as to which objective between language modeling and the Cloze task is better for the end performance. The best setup for each task improves the baseline by up to +1.0% Bleu and -1.1% Ter in the Romanian→English and +1.2% Bleu and -1.4% Ter in the Turkish→English.

Encoder Pre-Training vs. Decoder Pre-Training. Table 4.10 highlights the importance of the NMT model components in monolingual pre-training. For both monolingual objectives, the best results are obtained by pre-training both the encoder and the decoder, as expected. We achieve a similar performance by pre-training only the encoder, but when we train only the decoder, it gives only marginal improvements or even a slight degradation over the baseline. This reveals that the monolingually pre-trained encoder parameters are more useful than the decoder for translation purposes. A possible reason could be that the decoder pre-training step learns only the target self-attention parameters and ignores the encoder-decoder attention parameters. The decoder self-attention deals only with the fluency based on the target history, but the encoder-decoder attention actually connects the source language to the target language, which is the core part of translation. Even after pre-training the self-attention, the whole decoder needs to

Table 4.10: Ablation study on monolingual pre-training of different NMT model components (WMT 2017 Turkish→English). All experiments except the baselines follow the sequential transfer scheme.

Data Size (#sents)			Encoder Monolingual Objective	Decoder Monolingual Objective	`newstest2016`		`newstest2017`	
tr-en	tr	en			Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
207k	-	-	-	-	19.0	70.5	18.9	71.1
207k	4.8M	-	LM	-	19.6	69.5	19.4	69.9
	-	55M	-	LM	19.2	70.6	19.1	70.8
	4.8M	55M	LM	LM	19.6	70.1	19.4	69.8
	4.8M	-	Cloze	-	19.9	68.8	19.6	69.7
	-	55M	-	Cloze	19.2	70.3	18.7	70.7
	4.8M	55M	Cloze	Cloze	**20.0**	**68.5**	**19.8**	**69.2**

be retrained anyhow to properly use the source sentence information. In contrast, all of the parameters in the encoder can be pre-trained together and reused with smaller adjustments, which are responsible only for representing the source input.

Monolingual Pre-Training vs. Cross-lingual Pre-Training. Table 4.11 shows the effect of cross-lingual training (Section 4.4.2) for monolingual objectives. With a language modeling objective, cross-lingual pre-training is clearly better than separate pre-training of the encoder/decoder for the corresponding language. Cross-lingual training outperforms the ordinary monolingual training when using the Cloze task objective in the Romanian→English task, but underperforms in the Turkish→English task. All in all, despite the slight deviation, we show that cross-lingual training can be a viable option to further improve the usage of monolingual objectives for NMT.

Effect in Low-Resource Tasks. Intuitively, exploiting additional monolingual data would be more effective when bilingual training data is less available. To investigate this hypothesis, low-resource scenarios were artificially created by randomly selecting 200k, 60k, and 20k sentence pairs from the Romanian→English training data. As we look from right to left on the curves of Figure 4.7, the improvement resulting from monolingual pre-training is larger as the bilingual data decreases. In the lowest-resource setting, the sequential transfer with the cross-lingual Cloze task modeling has a gain of +7.5% Bleu against the baseline. The sequential transfer performance for 20k bilingual sentences is on par with the baseline for 60k bilingual sentences, and the same holds analogously between 60k- and 200k-pair conditions.

4.5 Comparison among the Presented Methods

LM integration (Section 4.2), synthetic data generation (Section 4.3), and LM/Cloze task training (Section 4.4) all leverage monolingual data in NMT. Each previous section includes the experimental results and analyses of each corresponding concept. In this section, we conduct a systematic comparison of the three concepts in consistent data settings where the same target monolingual data is available. This is done in the two previously employed tasks: WMT 2017 Turkish→English (Section B.1) and WMT 2018 German→English (Section B.2).

Table 4.12 reports the performance of each of the three methods. When each technique is applied alone, synthetic bilingual data is the most effective semi-supervised method, which provides additional training data in the exact form that the model expects. Between the other two

Table 4.11: Effect of cross-lingual training with monolingual objectives for NMT. All experiments except the baselines follow the sequential transfer scheme.

(a) WMT 2016 Romanian→English task

Data Size (#sents)			Monolingual Objective	Training Strategy	`newsdev2016`		`newstest2016`	
ro-en	ro	en			BLEU [%]	TER [%]	BLEU [%]	TER [%]
612k	-	-	-	-	35.5	51.3	33.7	53.7
612k	2.3M	55M	LM	Monolingual	36.6	50.1	34.7	52.8
				Cross-lingual	36.7	50.3	35.0	52.2
			Cloze	Monolingual	36.3	50.4	34.7	52.6
				Cross-lingual	**36.7**	**50.1**	**35.1**	**52.3**

(b) WMT 2017 Turkish→English task

Data Size (#sents)			Monolingual Objective	Training Strategy	`newstest2016`		`newstest2017`	
tr-en	tr	en			BLEU [%]	TER [%]	BLEU [%]	TER [%]
207k	-	-	-	-	19.0	70.5	18.9	71.1
207k	4.8M	55M	LM	Monolingual	19.6	70.1	19.2	70.3
				Cross-lingual	20.0	69.3	19.7	69.4
			Cloze	Monolingual	**20.0**	**68.5**	**19.8**	**69.2**
				Cross-lingual	19.8	69.1	19.4	69.8

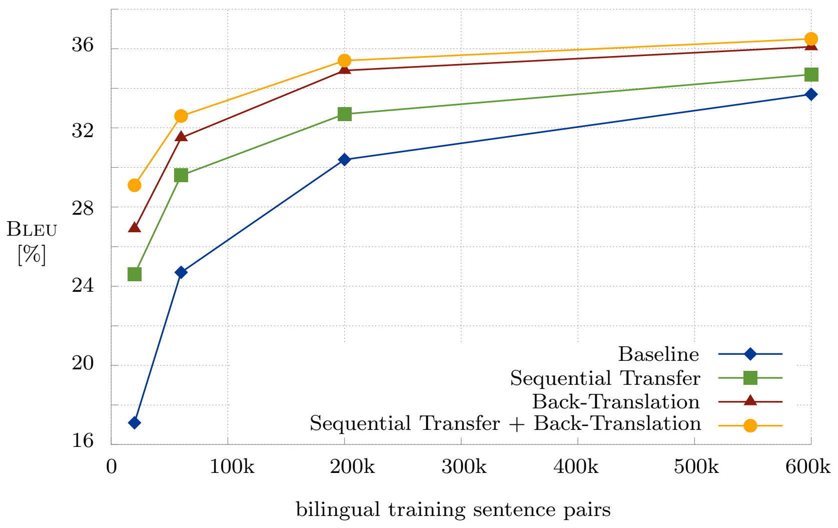


Figure 4.7: Sequential transfer performance with varying sizes of bilingual training data (WMT 2016 Romanian→English `newstest2016`). The pre-training was done for the Cloze task in a cross-lingual fashion. Adapted from [Nix 19].

methods, exploiting a part of the model parameters in a different form (monolingual training) is more beneficial than attaching an extra model component (LM integration). The best practice for semi-supervised NMT is combining all three techniques, although monolingual training or LM integration has only a marginal impact on the performance already improved by synthetic data.

Table 4.12: Comparison of semi-supervised methods for NMT given additional target monolingual data.

(a) WMT 2017 Turkish→English

Data Size (#sents)			`newstest2016`		`newstest2017`	
			Bleu	Ter	Bleu	Ter
tr-en	en	Method	[%]	[%]	[%]	[%]
207k	-	-	19.0	70.5	18.9	71.1
207k	100M	LM integration	20.2	69.1	19.5	70.1
		Synthetic data	24.8	64.8	23.6	66.3
		+ LM integration	25.0	64.4	23.6	66.4
		Sequential transfer	20.0	68.5	19.8	69.2
		+ Synthetic data	**25.1**	**64.2**	**23.7**	**66.2**
		+ LM integration	25.1	64.4	23.6	66.5

(b) WMT 2018 German→English

Data Size (#sents)			`newstest2017`		`newstest2018`	
			Bleu	Ter	Bleu	Ter
de-en	en	Method	[%]	[%]	[%]	[%]
5.9M	-	-	32.6	53.4	39.2	46.1
5.9M	100M	LM integration	33.5	53.8	40.1	46.4
		Synthetic data	36.5	49.7	43.0	42.0
		+ LM integration	36.6	49.8	43.2	41.9
		Sequential transfer	34.2	52.0	41.3	44.8
		+ Synthetic data	**36.7**	**49.3**	**43.8**	**40.5**
		+ LM integration	36.7	49.4	43.9	40.7

4.6 Comparison to Other Work

This section compares the outcomes reported in this chapter to other work on semi-supervised NMT, including international evaluation campaign submissions and recent publications. The comparison demonstrates where this thesis work stands in the MT community and how follow-up research has been developed. Table 4.13 shows the best results from this chapter and some of the noticeable work from others. Note that all results in each subtable are comparable in the sense that they are computed for the same test set with the same scoring script. Unavoidably, the data condition, preprocessing steps, model specification, and training hyperparameters are not exactly identical over different works, although they are chosen to be as close as possible to the experimental setting of this thesis.

In the Turkish→English task, our result outperforms the winning submission for WMT 2017, which is based on RNN models and a much smaller amount of monolingual data. The WMT 2018 winner has the closest setup to our experiments, except they filtered synthetic bilingual data with respect to the test domain and used up to 8 GPUs at the same time in training. They report a large improvement upon our best result, where we used only a single GPU without any data filtering. This shows the importance of domain adaptation and a larger batch size in training.

The lower three rows are from research papers which evaluate their semi-supervised approaches on this task. Their baselines are far behind the practical standard of NMT system building these days, which may be sufficient for a scientific proof of the effect of additional techniques but leave doubts in their applicability. In the end, their best results are all inferior to this thesis work, sometimes even worse than our baseline result (Table 4.12a).

Table 4.13: Comparison to other work on semi-supervised NMT.

(a) WMT 2017 Turkish→English

Data Size (#sents)			newstest2017
tr-en	en	Work	Bleu [%]
207k	100M	This thesis	**23.6**
207k	400k	WMT 2017 winner [Sennrich & Birch+ 17]	20.1
	10M	WMT 2018 winner [Deng & Cheng+ 18]	27.0
207k	3M	LM simple fusion [Stahlberg & Cross+ 18]	17.0
	30M	LM prior [Baziotis & Haddow+ 20]	18.5
	800k	Dual reconstruction [Xu & Niu+ 20]	19.4

(b) WMT 2018 German→English

Data Size (#sents)			newstest2017
de-en	en	Work	Bleu [%]
5.9M	100M	This thesis	**36.7**
5.9M	10M	WMT 2017 winner [Sennrich & Birch+ 17]	35.1
24M	18M	WMT 2018 winner [Schamper & Rosendahl+ 18]	38.3
4.5M	4.5M	Back-translation targeted sampling [Fadaee & Monz 18]	32.1
	5.0M	Dual reconstruction [Xu & Niu+ 20]	29.1
5.9M	12M	Iterative back-translation [Hoang & Koehn+ 18]	36.1
	193M	Combination with noisy channel [Yee & Dauphin+ 19]	36.2

Also in the German→English task, the result of this chapter is outstanding among other works, except the most comparable system of the WMT 2018 winner, which used a lot more bilingual data, multiple GPUs, and domain-specific filtering.

4.7 Conclusion and Contributions

In this chapter, we investigated three semi-supervised methodologies for NMT. Firstly, we empirically confirmed that integrating an LM into NMT is considerably effective if the search hyperparameters are correctly tuned, showing e.g. from 19.0% to 20.3% Bleu in WMT Turkish→English `newstest2016`. We verified that LM PPL does not represent improvement through integration, showing that count-based LMs give comparable results to neural LMs. We explored a novel position-level combination of an LM and an NMT model, although its performance does not differ significantly from the conventional sentence-level combination.

Secondly, we investigated various alternatives to beam search in generating synthetic bilingual data. By quantitative and qualitative analyses, we found out that the traditional beam search prefers frequent words too much and produces synthetic examples that are too monotonic. As a remedy, we proposed τ-restricted sampling and N-best list sampling, where the former method was proved to be the best and most consistent synthetic data generation strategy for NMT, showing e.g. from 32.6% to 36.5% Bleu in WMT German→English `newstest2017`. Compared to [Edunov & Ott+ 18], we showed that sampling without restriction might lead to detrimental synthetic data, depending on the language pair and test sets.

Lastly, we examined training NMT Transformer model components with monolingual objectives, achieving e.g. from 33.7% to 35.1% Bleu in WMT Romanian→English `newstest2016`.

We verified that, for the final translation performance, sequential transfer is more effective than multi-task learning, and language modeling and the Cloze task modeling are equally effective. In addition, we proposed and confirmed the impact of cross-lingual pre-training. The monolingual training for NMT was shown to have more impact in lower-resource tasks.

Among the three methodologies, synthetic bilingual data performs the best and its combination with other methods gives little or no additional improvements. In the following, individual contributions and public outcomes are summarized.

Language Model Integration. Section 4.2 is based on the first part of the Master's thesis of Miguel Graça [Graça 19], which was supervised by Yunsu Kim, Julian Schamper, and Shahram Khadivi. Julian Schamper initiated the topic and was involved in designing the main experiments. Yunsu Kim directed the research, planned and verified the datasets and experiments, together with Julian Schamper. Miguel Graça formulated the details of the position-level combination (Section 4.2.2), implemented all presented methods and conducted all experiments.

Synthesizing Bilingual Data. Section 4.3 is based on the second part of Miguel Graça's Master's thesis [Graça 19], whose core findings are extended and published in WMT 2019 [Graça & Kim$^+$ 19]. Yunsu Kim and Julian Schamper proposed the topic and devised τ-restricted sampling and N-best list sampling. Miguel Graça unified the generation strategies and formulated the cross-entropy criterion. He also implemented all generation strategies and designed and conducted all experiments, except that the scaled comparisons of the strategies (Figure 4.4) were done by Yunsu Kim. Shahram Khadivi supported discussions and interpretations of the experimental results.

Monolingual Training. Section 4.4 is based on the Master's thesis of Arne Nix [Nix 19], which was supervised by Jan Rosendahl, Yunsu Kim, and Shahram Khadivi. Jan Rosendahl and Julian Schamper started the idea of LM training for Transformer NMT models. Yunsu Kim and Jan Rosendahl led the whole research, including data preparation, main experiment designs and analysis of the results. Arne Nix proposed cross-lingual training, implemented all presented techniques, and ran most of the experiments. The cross-lingual LM pre-training experiments were conducted by Yunsu Kim. Shahram Khadivi was involved in all discussions and decisions about which experiments should be done.

The combination of different semi-supervised methodologies (Section 4.5) was performed by Yunsu Kim.

5. Cross-lingual Learning

This chapter examines low-resource NMT methods utilizing bilingual corpora of language pairs other than the original task. These methods are based on the fact that translation tasks for different languages, e.g. German→English and French→English, are not completely separate from each other but may be closely related. They all analyze and generate sentences in human languages, which have commonalities in syntax, morphology, alphabet, etc. to different degrees. For example, German, French, and English are all European languages and use Latin alphabet letters, sharing many word stems and general sentence structures. Therefore, between German→English and French→English tasks, one can find plenty of similar translation/reordering rules that correspond word by word.

This cross-lingual relation of translation tasks can be readily implemented with neural network models that learn intermediate representations in continuous-valued vector spaces without mapping vocabulary words [Johnson & Schuster$^+$ 17]. As stated in Section 4.4, by sharing the representation space among different domains of inputs/outputs, training neural network models for multiple tasks is shown to be beneficial due to the implicit interaction among the tasks and the task-oriented regularization. Technically, for cross-lingual NMT, this means reusing model parameters of other language pairs (parent) in the final translation task (child) with some special care for dissimilar vocabularies. A parent language and a child language are *related* if they belong to the same language family, otherwise we call them *distant* or *unrelated*.

Depending on the data condition of the child task, we divide cross-lingual learning problems of NMT into three categories:

- *Low-resource*: given a small amount of bilingual corpora for child training
- *Zero-resource*: given no bilingual corpora for the child, but usually meaning that we artificially generate a bilingual corpus for the child task and use it for child training
- *Zero-shot*: a subcategory of the zero-resource problem, where bilingual training corpora are not used at all for child training

Note that in the zero-shot problem, we never learn to perform the child task but still check the performance on the child task, where our model relies solely on the knowledge of parents.

This chapter has three main sections, each of which deals with a different transfer scheme:

- Joint training for parents and children together (Sections 5.2.1 and 5.3.1)
- Pre-training for parents and then fine-tuning for children (Sections 5.2.2 and 5.3.2)
- Concatenating pre-trained parent models to build a child model (Section 5.3.3)

where each scheme is concretized for either or both of the two transfer scenarios:

- The language is switched on one side from a parent (e.g. parent: German→English, child: Basque→English; Section 5.2)

- The language is switched on either side from multiple parents (e.g. parents: French→English and English→German, child: French→German; Section 5.3)

We consider each concrete method in the three problems categorized above: low-resource, zero-resource, and zero-shot.

5.1 Related Work

Transfer Learning and Multilinguality. The idea of building a common representation of languages was introduced around 50 years ago using the term *interlingua* [Vauquois 68]. However, it has never been successful since it assumes the interlingua to be another language with discrete entities (*meta-language*), which must be very complex to accommodate mapping from multiple natural languages.

For NMT, the easiest case of transfer learning is across text domains. Having an NMT model trained on some data, we can continue the training from the same network parameters with data from another domain [Luong & Manning 15, Freitag & Al-Onaizan 16]. Transfer from another natural language processing task is also straightforward; for example, we can initialize the parameters of NMT models with pre-trained language models of corresponding languages, since the encoder and decoder are essentially language models except for a few additional translation-specific components [Ramachandran & Liu$^+$ 17, Conneau & Lample 19].

Multilingual sharing and joint training of NMT model parameters is first proposed in [Dong & Wu$^+$ 15]. They train the same English encoder for multiple from-English language pairs (*one-to-many*), having a separate decoder with its encoder-decoder attention component for each task. [Firat & Cho$^+$ 16] extend the concept to *many-to-many* translation, where each source language is modeled by a single encoder and each target language by a single decoder, regardless of the language pairs. They share the encoder-decoder attention component across all language pairs to improve the connectivity between encoders and decoders. [Lee & Cho$^+$ 17] train a single encoder for multiple source languages using a joint vocabulary at the character level, implementing *many-to-one* translation. [Johnson & Schuster$^+$ 17] finally share the entire NMT model components across multiple source and target languages, presenting a single model for many-to-many translation. They also show a reasonable performance on zero-shot translation for the first time. [Aharoni & Johnson$^+$ 19] scale up this single-model approach to handle around 100 languages.

A problem of a single encoder/decoder for multiple languages is that it relies on shared subword vocabularies so it is hard for their model to adapt to a new language. [Platanios & Sachan$^+$ 18] augment a multilingual model with language-specific embeddings from which the encoder and decoder parameters are inferred with additional linear transformations. They only mention its potential to transfer to an unseen language but do not present any results on it. [Wang & Pham$^+$ 19] address the vocabulary mismatch in multilingual NMT by using shared embeddings of character n-grams and common semantic concepts. Their method has a strict assumption that the languages should be related orthographically with shared alphabets.

As opposed to joint training, [Zoph & Yuret$^+$ 16] show sequential transfer for NMT, optimizing the model explicitly for a child task given a pre-trained parent model. This prevents the model from losing its capacity to process parent languages while utilizing the information from related language pairs well. However, this scheme is also ineffective or inefficient for many transfer scenarios, especially when the parent and the child are unrelated. [Nguyen & Chiang 17] and [Kocmi & Bojar 18] use shared subword vocabularies to improve the transfer for more languages and help target language switches, but this requires retraining of the parent model whenever we transfer to a new child language. [Neubig & Hu 18] improve the transfer with additional bilingual data of a language pair that is similar to the child, relying on the relatedness of the languages. They learn just a separate subword vocabulary for the child language without further care. Section 5.2.2

shows effective techniques to transfer a pre-trained NMT model to a new, unrelated language without shared vocabularies.

Non-English Translation Tasks. For the case of transfer from multiple parents, non-English language pairs are common testing scenarios. To improve a source→target child, one can leverage a pivot language by having source→pivot and pivot→target parents, where the pivot language is usually English.

The most naive approach is pivoting (pivot translation), i.e. reusing (already trained) source→ pivot and pivot→target models directly, decoding twice via the pivot language [Kauers & Vogel$^+$ 02, De Gispert & Marino 06]. One can keep the N-best hypotheses in the pivot language to reduce the prediction bias [Utiyama & Isahara 07] and improve the final translation by system combination [Costa-Jussà & Henríquez$^+$ 11]. This, however, increases the translation time even more. In multilingual NMT, [Firat & Sankaran$^+$ 16] modify the second translation step (pivot→target) to use source and pivot language sentences together as the input.

We may also create pivot-based synthetic bilingual data by translating the pivot side of given pivot-target bilingual data using a pivot→source model [Bertoldi & Barbaiani$^+$ 08], or the other way around, translating source-pivot data using a pivot→target model [De Gispert & Marino 06]. For NMT, the former is extended by [Zheng & Cheng$^+$ 17] to compute the expectation over synthetic source sentences. The latter is also called the teacher-student approach [Chen & Liu$^+$ 17], where the pivot→target model (teacher) produces target hypotheses for training the source→target model (student).

Lastly, model training can also be modified particularly for the pivot-based scenarios. In phrase-based MT, there have been many efforts to combine phrase/word-level features of source-pivot and pivot-target into a source→target system [Utiyama & Isahara 07, Wu & Wang 07, Bakhshaei & Khadivi$^+$ 10, Zahabi & Bakhshaei$^+$ 13, Zhu & He$^+$ 14, Miura & Neubig$^+$ 15]. In NMT, [Cheng & Yang$^+$ 17] jointly train for three translation directions of source-pivot-target by sharing network components, while [Ren & Chen$^+$ 18] use the expectation-maximization algorithm with the target sentence as a latent variable. [Lu & Keung$^+$ 18] deploy intermediate recurrent layers which are common for multiple encoders and decoders, while [Johnson & Schuster$^+$ 17] share all components of a single multilingual model. Both methods train the model for language pairs involving English but enable zero-shot translation for unseen non-English language pairs. For this, [Ha & Niehues$^+$ 17] encode the target language as an additional embedding and filter out non-target tokens in the output. [Lakew & Lotito$^+$ 17] combine multilingual training with synthetic data generation to improve the zero-shot performance iteratively, while [Sestorain & Ciaramita$^+$ 18] apply the NMT prediction score and a language model score to each synthetic example as gradient weights.

Section 5.3.2 presents novel pre-training and transfer techniques for a non-English language pair, and Section 5.3.3 extends the pivoting with integrated training.

5.2 To-English Tasks

Cross-lingual transfer from a translation task (parent) to another (child) is easier when either source or target language is common in both the parent and the child tasks. The common language plays a role as an anchor, where many parts of the parent parameters regarding that language can be directly usable for the child task. This constraint is often easily achievable between language pairs involving English.

This section focuses on the tasks where the target language is fixed to English. English is the most popular language in the world, and bilingual datasets are mostly available for language pairs for which either the source or the target side is English. For a low-resource to-English task, e.g. Basque→English, it is straightforward to find a high-resource parent task from which one can positively consider cross-lingual transfer. This scenario represents the case that knowledge and opinions in a minority language are made understandable worldwide.

5.2.1 Multilingual Model: Many-to-One

The first cross-lingual methodology we cover in this thesis is the *multilingual* NMT model, i.e. a joint model that can perform multiple translation tasks for different language pairs. Its goal is to show reasonable performance on all associated tasks, so every task can be seen as a child. On the other hand, in training, each task transfers its knowledge to other tasks within the same model parameters, which is the role of a parent. Thus there is no distinction between parents and children in a multilingual model. The concept originates from multi-task learning [Caruana 97], where, in the context of MT, each task corresponds to a different translation direction. It has been advocated due to its deployment efficiency, where one model can handle various types of user input.

To build a multilingual model, we optimize for all desired language pairs together from the beginning of the training. Since the data size may vary wildly across multiple translation tasks, it is important to maintain the balance among the tasks in training to avoid overoptimization for high-resource language pairs. This can be done by alternating mini-batches of different language pairs [Dong & Wu$^+$ 15] or assembling each mini-batch from training examples of all the language pairs, keeping the amount of training signal almost equal over all tasks. The development set should also contain examples of all the language pairs to ensure balance in the performance. If an encoder or a decoder is shared among multiple languages, it is necessary to build a joint vocabulary of all languages on the input or output side, respectively. Such a vocabulary is usually constructed at the subword or character level to increase the overlapping tokens and make the word embedding matrix sufficiently small.

Depending on the number of languages to model on each side, the main architecture types of multilingual models are categorized into many-to-one, one-to-many, or many-to-many. We assume a single encoder for all input languages and a single decoder for all output languages, which is the most recent and effective sharing decision at the time of writing this thesis.

If the model learns to translate from multiple source languages to one target language, it is called a *many-to-one* scheme. The common target language is usually English, since most bilingual resources are for a from-/to-English language pair. Figure 5.1 shows a simplified diagram of a many-to-one model. The goal here is to make the encoder language-agnostic so that the decoder can generate a decent target translation no matter which source language the input is given in. This means that the encoder should have good *cross-linguality*, i.e. the representation space is shared effectively among multiple input languages.

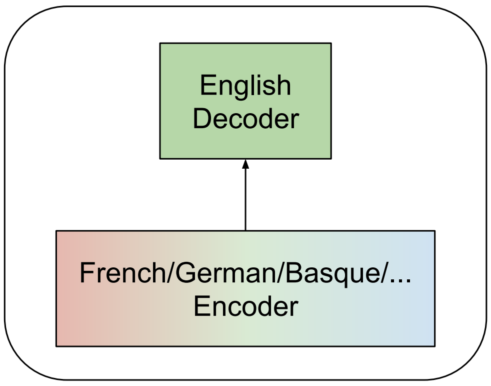


Figure 5.1: Many-to-one multilingual model (single encoder/decoder).

5.2.2 Sequential Transfer

For a multilingual single model to work in practice, we need to preprocess bilingual datasets of multiple language pairs jointly before training. This has two critical issues:

1. The languages for training should be linguistically related in order to build a shared vocabulary.
2. It is not feasible to add a new language to a trained model, since the training vocabulary must be redefined; one may need to retrain the model from scratch.

Sequential transfer is an alternative where an NMT model is first pre-trained on a high-resource language pair (parent) and then fine-tuned with bilingual data of another language pair (child) [Zoph & Yuret$^+$ 16]. There is a non-cyclic order of translation tasks for which the model is trained, which makes a clear distinction between a parent and a child. Instead of the interaction in multilingual models, the parent of sequential transfer makes a one-way contribution: a good initialization for the child. At the end of the training, the model expects to perform well only on the child task. The performance on the parent task is not measured; the parent is supposed to only help the child's performance.

Adapting to a new language is conceptually simpler in sequential transfer. When fine-tuning for a child, we are allowed to ignore the parent vocabulary and optimize the model with the new vocabulary of the child. Thus there is no need to restructure a joint vocabulary and retrain from scratch accordingly. Here, an MT model for each language pair is maintained, so we do not have to worry about the degradation of performance in high-resource language pairs (parents).

In this section, we study the simplest and most straightforward NMT transfer when there is only one high-resource parent task, especially for to-English language pairs. Figure 5.2 shows an example of the single-parent transfer. The target language is fixed to English for both the parent and the child, and the transfer of the decoder parameter is relatively smooth. On the language-switching side (source), the transfer is much more difficult due to the language dissimilarity between the parent and the child in e.g. alphabet, morphology, sentence structure, etc. This might introduce a completely different data space that does not fit to the pre-trained model. It becomes more severe when the parent and child source languages are from distinct language families.

In the following, we present such discrepancies in detail and propose correpsonding solutions. In short,

- We alleviate the vocabulary mismatch between parent and child languages via cross-lingual word embedding.

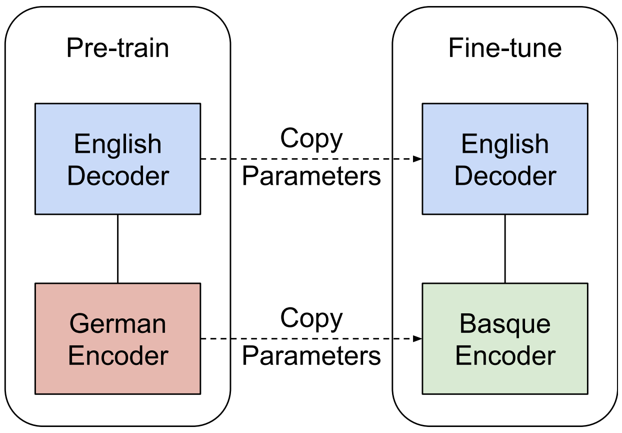


Figure 5.2: Example of single-parent transfer from German→English to Basque→English.

- We train a more general encoder in the parent training by injecting artificial noise, making it easier for the child model to adapt to it.
- We generate synthetic data from bilingual data of the parent language pair, improving the low-resource transfer where the conventional back-translation [Sennrich & Haddow+ 16b] fails.

Different Vocabulary: Cross-lingual Word Embedding

The biggest challenge of cross-lingual transfer is the vocabulary mismatch between the transferrer (parent) and the transferree (child). A natural language vocabulary is discrete and unique for each language, while the mapping between two different vocabularies is non-deterministic and arbitrary. Therefore, when we merely replace a source language, the NMT encoder will see totally different input sequences; pre-trained encoder weights do not get along with the source embedding anymore. This might seriously limit the performance, especially for transfer among distant languages.

A popular solution to this is sharing the vocabulary among the languages of concern [Nguyen & Chiang 17, Kocmi & Bojar 18]. This is often implemented with joint learning of subword units [Sennrich & Haddow+ 16c]. Despite its effectiveness, it has an intrinsic problem in practice: A parent model must already be trained with a shared vocabulary with the child languages. Such a pre-trained parent model can be transferred only to those child languages using the same shared vocabulary. When we adapt to a new language whose words are not included in the shared vocabulary, we need to learn a joint subword space again with the new language and retrain the parent model accordingly—very inefficient and not scalable. A shared vocabulary is also problematic in that it must be divided into language-specific portions. When many languages share it, the allocated portion for each will be smaller and accordingly less expressive. This is the reason why the vocabulary is usually shared only for linguistically related languages, effectively increasing the portion of common surface forms.

In this thesis, we propose to keep the vocabularies separate, but share their embedding spaces instead of surface forms. This can be done independently from the parent model training and requires only monolingual data of the child language:

1. Learn monolingual embedding of the child language $\mathrm{b}_f^{\mathrm{c}}$, using e.g. the skip-gram algorithm [Mikolov & Chen+ 13].
2. Extract source embedding $\mathrm{b}_f^{\mathrm{p}}$ from a pre-trained parent NMT model.
3. Learn a cross-lingual linear mapping $\mathbf{W}_{\mathrm{c,p}} \in \mathbb{R}^{D\times D}$ between 1 and 2 by minimizing the objective below:

$$\sum_{(f,f')\in\mathcal{D}_{\mathrm{p,c}}} \|\mathbf{W}_{\mathrm{c,p}} \cdot \mathrm{b}_f^{\mathrm{c}}(f) - \mathrm{b}_f^{\mathrm{p}}(f')\|_2 \tag{5.1}$$

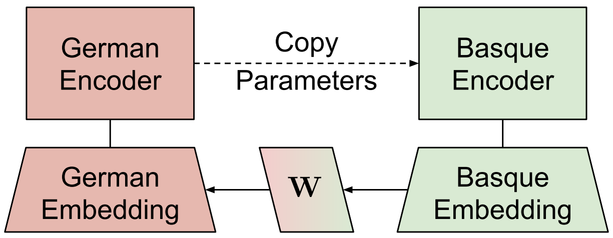


Figure 5.3: Cross-lingual mapping of a child (Basque) embedding to the parent (German) embedding.

4. Replace source embedding of the parent model parameters with the learned cross-lingual embedding.

$$\mathrm{b}_f^{\mathrm{p}} \leftarrow \mathbf{W}_{\mathrm{c,p}} \cdot \mathrm{b}_f^{\mathrm{c}} \tag{5.2}$$

5. Initialize the child model with 4 and start the NMT training on the child language pair.

The dictionary $\mathcal{D}_{\mathrm{p,c}}$ in Step 3 can be obtained in an unsupervised way by adversarial training [Conneau & Lample+ 18] or matching digits between the parent and child languages [Artetxe & Labaka+ 17]. The mapping $\mathbf{W}_{\mathrm{c,p}}$ can also be iteratively refined with self-induced dictionaries of mutual parent-child nearest neighbors [Artetxe & Labaka+ 17], which is still unsupervised. The cross-lingually mapped child embeddings fit better as input to the parent encoder, since they are adjusted to a space similar to that of the parent input embeddings (Figure 5.3). Note that in Step 4, the mapping $\mathbf{W}_{\mathrm{c,p}}$ is not explicitly inserted as additional parameters in the network. It is multiplied by $\mathrm{b}_f^{\mathrm{c}}$ and the result is used as the initial source embedding weights. The initialized source embedding is also fine-tuned along with the other parameters in the last step.

These steps do not involve rearranging a joint vocabulary or retraining of the parent model. Using this method, one can pre-train a single parent model once and transfer it to many different child languages efficiently. This method is also effective for non-related languages that do not share surface forms, since we address the vocabulary mismatch at the embedding level. After each word has been converted to its embedding, it is just a continuous-valued vector in a mathematical space; matching vocabularies is done by transforming the vectors irrespective of language-specific alphabets. Note that this method does not involve any change in the model architecture. Also, it is not limited to the transfer between similar languages and directly benefits from advances in cross-lingual word embedding for distant languages.

Cross-lingual word embedding is studied for the use in MT as follows. In phrase-based SMT, [Alkhouli & Guta+ 14] build translation models with word/phrase embeddings. [Kim & Geng+ 18] use cross-lingual word embedding as a basic translation model for unsupervised MT and attach other components on top of it. [Artetxe & Labaka+ 18] and [Lample & Denoyer+ 18] initialize their unsupervised NMT models with pre-trained cross-lingual word embeddings. [Qi & Sachan+ 18] do the same initialization for supervised cases, observing improvements only in multilingual setups.

Different Syntax: Noisy Pre-Training

Another main difference between languages is the word order, namely the syntactic structure of sentences. Neural sequence-to-sequence models are highly dependent on the sequential ordering of the input, i.e. absolute/relative positions of input tokens. When we train an encoder for a language, it learns the language-specific word order conventions, e.g. position of a verb in a clause, structure of an adverb phrase, etc. If the input language is changed, the encoder must adjust itself to unfamiliar word orders. The adaptation gets more difficult for non-related languages.

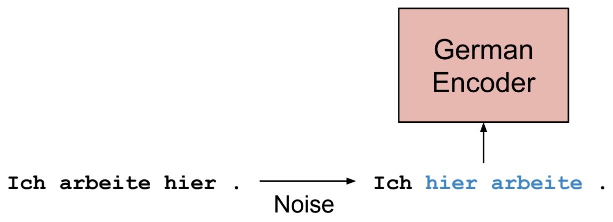
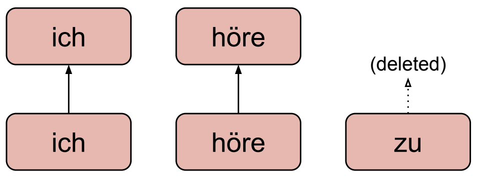
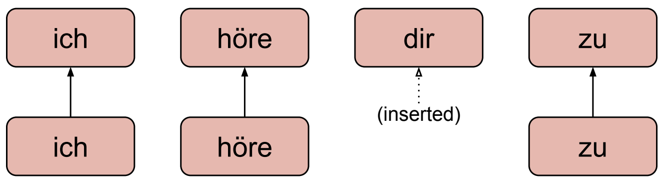


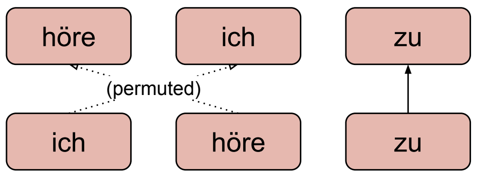
Figure 5.4: Injecting noise into a parent (German) source sentence.



(a) Deletion



(b) Insertion



(c) Permutation

Figure 5.5: Example for each noise type for the noisy pre-training of a parent model.

To mitigate this syntactic difference in cross-lingual transfer for NMT, we suggest generalizing the parent encoder so that it is not overoptimized to the parent source language. We achieve this by modifying the source side of the parent training data, artificially changing its word orders with random noise (Figure 5.4). The noise function includes [Hill & Cho$^+$ 16, Kim & Geng$^+$ 18]:

- Inserting a word between the original words uniformly with a probability p_i at each position, choosing the inserted word uniformly from the top V_i frequent words (Figure 5.5a)
- Deleting original words uniformly with a probability p_d at each position (Figure 5.5b)
- Permuting the original word positions uniformly within a limited distance d_p (Figure 5.5c)

The noises are injected into every source sentence differently for each epoch. The encoder then sees not only word orders of the parent source language but also other various sentence structures. Since we set limits to the randomness of the noise, the encoder is still able to learn a general monotonicity of natural language sentences. This makes it easier for the parent encoder to adapt to a child source language, effectively transferring the pre-trained language-agnostic knowledge of input sequence modeling.

Different Training Signal: Mixed Fine-Tuning with Parent Training Data

From the beginning of the fine-tuning, sequential transfer gives radically different training signals to the model compared to the pre-training stage. The abrupt transition of the training data is due not only to the language switch but also to the data size. In a transfer learning scenario, a child usually has a much smaller amount of training data, which changes the training objective dramatically and makes the model vulnerable to overfitting. For NMT, the standard technique to address the scarcity of training data is to generate synthetic bilingual data from target

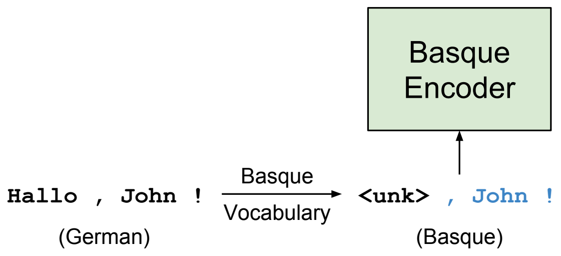


Figure 5.6: Example of a parent sentence (German) mapped to the child vocabulary for fine-tuning.

monolingual corpora (Section 4.3). However, this works only if the target-to-source translation model is of sufficiently acceptable quality, which is hard to train for low-resource child tasks.

For these scenarios, we devise a simple trick to create additional bilingual data for the child language pair without training a target-to-source translation model. The idea is to reuse the bilingual data already used for training the parent model. In the source side, we retain only those tokens that exist in the child vocabulary and replace all other tokens with a predefined token, e.g. `<unk>` (Figure 5.6). The target side stays the same as we do not switch the languages. The source side of this synthetic data consists only of the overlapping vocabulary entries between the parent and child languages. By including this data in the child model training, we prevent an abrupt change of the input to the pre-trained model while keeping the parent and child vocabularies separate. It also helps to avoid overfitting to the tiny bilingual data of the child language pair.

In addition, we can expect a synergy with cross-lingual word embedding (Section 5.2.2), where the source embedding space of the child task is transformed into that of the parent task. In this cross-lingual space, an overlapping token between parent and child vocabularies should have a very similar embedding to that in the original parent embedding space with which the pre-trained encoder is already familiar. This helps to realize a smooth transition from parent source input to child source input in the transfer process.

5.2.3 Experiments

We verify the effect of our techniques in transfer learning setups with five different child source languages: Basque (eu), Slovenian (sl), Belarusian (be), Azerbaijani (az), and Turkish (tr). The target language is fixed to English (en) and we use German→English as the parent language pair. The parent model was trained on bilingual data of the WMT 2018 news translation task (Section B.2) and synthetic data released by [Sennrich & Haddow$^+$ 16a]. For the child language pairs, we used IWSLT 2018 low-resource MT task data (Section B.4), IWSLT 2014 MT task data (Section B.5), TED talk data from [Qi & Sachan$^+$ 18] (Section B.6 and B.7). Note that the child source languages (Isolate, Slavic, Turkic families) are linguistically distant from the parent source (Germanic family). The source side was lowercased and the target side was frequent-cased.

For the transfer learning, the source and target embedding weights are not tied. Each source language was encoded with BPE codes learned via 20k merge operations, while the target language was encoded with 50k BPE merges. Dropout with a probability of 0.3 was applied to all layers except embeddings in both parent and child model trainings. Training was carried out with Sockeye [Hieber & Domhan$^+$ 17] with the default parameters. We stopped the training when perplexity on a validation set was not improving for 12 checkpoints. We set checkpoint frequency to 10,000 updates for the parent model and 1,000 updates for the child models. The parent model yields 39.2% Bleu on WMT German→English `newstest2016` test set.

Table 5.1: Effect of sequential transfer techniques in to-English translation tasks.

	eu-en		sl-en		be-en		az-en	
	Bleu	Ter	Bleu	Ter	Bleu	Ter	Bleu	Ter
System	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
Baseline	1.7	96.6	10.1	82.9	3.2	92.5	3.1	97.8
Multilingual with de-en	5.1	85.3	16.7	71.7	4.2	90.6	4.5	94.7
Transfer from de-en	4.9	92.2	19.2	68.4	8.9	86.0	5.3	97.6
+ Cross-lingual word embedding	7.4	82.4	20.6	66.0	12.2	79.4	7.4	89.4
+ Noisy pre-training	8.2	80.2	21.3	65.4	12.8	79.5	8.1	88.5
+ Mixed fine-tuning	**9.7**	**77.5**	**22.1**	**64.4**	**14.0**	**78.1**	**9.0**	**87.0**

To pre-train word embeddings, we used Wikimedia dumps[1] of timestamp 2018-11-01 for all child languages. From Wikimedia dumps, the actual articles were extracted first,[2] and were split into sentences using the StanfordCoreNLP toolkit [Manning & Surdeanu$^+$ 14]. The statistics of the monolingual corpora are given in Section B.4–B.7. Monolingual embeddings were trained with fasttext [Bojanowski & Grave$^+$ 17] with minimum word count 0. For learning the cross-lingual mappings, we ran 10 epochs of adversarial training and another 10 epochs of dictionary-based refinement using MUSE [Conneau & Lample$^+$ 18]. We chose the top 20k types as discriminator inputs and 10k as maximum dictionary rank.

For the noisy pre-training, we used these values for the noise model, following [Kim & Geng$^+$ 18]: $p_\mathrm{i} = 0.1$, $V_\mathrm{i} = 50$, $p_\mathrm{d} = 0.1$, and $d_\mathrm{p} = 3$. We empirically found that these values are also optimal for our purpose. The parent model trained with noise gives 38.2% Bleu in WMT German→English `newstest2016`: 1.0% worse than without noise.

For the mixed fine-tuning, we uniformly sampled 1M sentence pairs from the German→English bilingual data used for the parent training and processed them according to Section 5.2.2. The child model bilingual data was oversampled to 500k sentence pairs, making an overall ratio of 1:2 between the bilingual and synthetic data. We also tried other ratio values, e.g. 1:1, 1:4, or 2:1, but the performance was consistently worse.

As a baseline child model without transfer learning, we used the same setting as above but learned a shared source-target BPE vocabulary with 20k merge operations. We also tied source and target embeddings as suggested for low-resource settings in [Schamper & Rosendahl$^+$ 18]. Dropout was also applied to the embedding weights for the baselines. We also compare our transfer learning with many-to-one multilingual models. To build a multilingual model, we learned a joint BPE vocabulary of the parent source, the child source, and target language with 32k merge operations. The training data for the child task was oversampled so that each mini-batch has roughly 1:1 ratio of the parent/child training examples. Note that we built a different multilingual model for each child task. Since they depend on shared vocabularies, we need to restructure the vocabulary and retrain the model for each of the new language pairs we wish to adapt to.

Main Results. Table 5.1 presents the main results. Plain transfer learning already gives a boost but is still far from a satisfying quality, especially for Basque→English and Azerbaijani→English. On top of that, each of our three techniques offers clear, incremental improvements in all child language pairs, with a maximum of 5.1% Bleu in total.

Cross-lingual word embedding shows a huge improvement up to +3.3% Bleu, which exhibits the strength of connecting parent-child vocabularies on the embedding level. If we train the parent model with artificial noise on the source side, the performance consistently increases by up to

[1]https://dumps.wikimedia.org/
[2]https://github.com/attardi/wikiextractor/

+0.8% Bleu. This occurs even when dropout is used in the parent model training; randomizing word orders provides meaningful regularization which cannot be achieved via dropout. Finally, using parent bilingual data in mixed fine-tuning proves to be effective in low-resource transfer to substantially different languages: We obtain an additional gain of at most +1.5% Bleu.

Our results also surpass the multilingual joint training by a large margin in all tasks. One shared model for multiple language pairs inherently limits the modeling capacity for each task. Particularly, if one language pair has much smaller training data than the other, oversampling the low-resource portion is not enough to compensate for the scale discrepancy in multilingual training. Transfer learning with our add-on techniques is more efficient at exploiting knowledge of high-resource language pairs and fine-tuning the performance towards a child task. In the following, we further investigate our methods in detail in comparison to their similar variants, and also perform ablation studies for the NMT transfer in general.

Types of Pre-trained Embedding. We analyze the effect of the cross-linguality of pre-trained embeddings in Table 5.2. We observe that monolingual embedding without a cross-lingual mapping also improves the transfer learning, but is significantly worse than our proposed embedding, i.e. mapped to the parent source (de) embedding. The mapping can also be learned with the target (en) side using the same procedure as in Section 5.2.2. The target-mapped embedding is not compatible with the pre-trained encoder but directly guides the child model to establish the connection between the new source and the target. It also improves the system, but our method is still the best among the three embedding types.

Table 5.2: Azerbaijani→English translation results with different types of pre-trained source embeddings.

Pre-trained Embedding	Bleu [%]	Ter [%]
None	5.3	97.6
Monolingual	6.3	92.4
Cross-lingual (Azerbaijani→German)	**7.4**	**89.4**
Cross-lingual (Azerbaijani→English)	7.1	89.6

Synthetic Data Generation. In Table 5.3, we compare our technique in Section 5.2.2 with other methods of generating synthetic data. For a fair comparison, we used the same target side corpus (1M sentences) for all these methods. As explained in Section 5.2.2, back-translation [Sennrich & Haddow+ 16b] is not beneficial here because the generated source is of too low quality. An empty source sentence is proposed along with back-translation as its simplification, which does not help either in transfer learning. Copying target sentences to the source side is yet another easy way to obtain synthetic data [Currey & Barone+ 17]. This gives an improvement to a certain extent; however, our method of using the parent model data works much better in transfer learning.

Table 5.3: Basque→English translation results with synthetic data generated using different methods.

Synthetic Data	Bleu [%]	Ter [%]
None	8.2	80.2
Back-translation	8.3	80.4
Empty source	8.6	79.3
Copied target	8.9	79.5
Parent model data	**9.7**	**77.5**
+ Cross-lingual replacement (German→Basque)	8.7	78.4

We manually looked at the survived tokens in the source side of our synthetic data. We observed lots of overlapping tokens over the parent and child source vocabularies even if they were not shared: 4,487 vocabulary entries between Basque and German. Approximately 2% of them are punctuation symbols and special tokens, 7% are digits, and 62% are made of Latin alphabets, a large portion of which is devoted to English words (e.g. named entities) or their parts. The rest of the vocabulary consists mostly of noisy tokens with exotic alphabets. As Figure 5.6 illustrates, just punctuation symbols and named entities can already define a basic structure of the original source sentence. Such tokens play the role of anchors in translation; they are sure to be copied to the target side. The surrounding `<unk>` tokens are spread according to the source language structure, whereas merely copying the target sentence to the source [Currey & Barone$^+$ 17] ignores the structural difference between source and target sentences. Note that our trick applies also to languages with completely different alphabets, e.g. Belarusian and German (see Table 5.1).

We also tested an additional processing for our synthetic data to reduce the number of unknown tokens. We replaced non-overlapping tokens in the German source side with the closest Basque token in the cross-lingual word embedding space. The result is, however, worse than not replacing them; we noticed that this subword-by-subword translation produces many Basque phrases with wrong BPE merges [Kim & Geng$^+$ 18].

Vocabulary Size. Table 5.4 estimates how large the vocabulary should be for the language-switching side in NMT transfer. We varied the number of BPE merges on the source side, fixing the target vocabulary to 50k merges. The best results are with 10k or 20k of BPE merges, which shows that the source vocabulary should be reasonably small to maximize the transfer performance. Fewer BPE merges lead to more language-independent tokens; it is easier for the cross-lingual embedding to find the overlaps in the shared semantic space. If the vocabulary is excessively small, we might lose too many language-specific details that are necessary for the translation process. This is shown in the 10k merges of Belarusian→English.

Table 5.4: Translation results with different sizes of the source vocabulary.

	sl-en		be-en	
BPE merges	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
10k	**21.0**	**65.4**	11.2	81.9
20k	20.6	66.0	**12.2**	**79.4**
50k	20.2	66.4	10.9	82.1
70k	20.0	66.5	10.9	81.8

Freezing Parameters. Lastly, we conducted an ablation study of freezing parent model parameters in the child training process (Table 5.5). We show only the results when freezing the decoder; in our experiments, freezing any component of the encoder always degrades the translation performance. The experiments were done at the final stage with all of our three proposed methods applied.

Target embedding and target self-attention parts are independent of the source information, so it makes sense to freeze those parameters even when the source language is changed. On the other hand, encoder-decoder attention represents the relation between source and target sentences, so it needs to be redefined for a new source language. The performance deteriorates when freezing feedforward sublayers, since it is directly influenced by the encoder-decoder attention layer. The last row means that we freeze all parameters of the decoder; it is actually better than freezing all but the output layer.

[Sachan & Neubig 18] show ablation studies on parameter sharing and freezing in a one-to-many multilingual setup with shared vocabularies. Our work conducts similar experiments in the

Table 5.5: Slovenian→English translation results with freezing different components of the decoder.

Frozen Parameters	Bleu [%]	Ter [%]
None	21.0	66.2
Target embedding	21.4	65.2
+ Target self-attention	**22.1**	**64.4**
+ Encoder-decoder attention	21.8	64.6
+ Feedforward sublayer	21.3	65.0
+ Output layer	21.9	64.5

transfer learning setting with separate vocabularies.

5.3 Non-English Tasks

MT research is biased towards language pairs including English due to the ease of collecting bilingual corpora. Non-English language pairs, e.g. French-German, neither of whose source or target side is English, have little or no bilingual available in most cases. Translation between non-English languages is thus usually done by pivoting through English, e.g. translating French (source) input to English first with a French→English model which is later translated to German (target) with an English→German model [De Gispert & Marino 06, Utiyama & Isahara 07, Wu & Wang 07]. Here, English in the middle is called the *pivot language*. We denote a pivot language sentence by $g_1^K = g_1, ..., g_k, ..., g_K$ with K as its length. The most common pivot language is English because of its popularity. Corpora including English are often large and are also tailored in translation quality and sentence diversity. While there are certainly scenarios where English might not be the best fit, in this thesis, all experiments use English as a pivot language.

However, vanilla pivoting has several clear drawbacks. First, it doubles the decoding time with two translation models. When the N-best list in the pivot language is considered, the decoding time could be even longer. This is problematic for many use cases where an instant translation between non-English languages is crucial, e.g. speech translation. Second, translation errors are propagated or expanded in the two-step process, which might leave important semantic content of the input out of the final output. Third, when given at least a small amount of bilingual data, there is no way to exploit it easily for improving the model performance. In this section, we attempt to go beyond plain pivoting, in order to find the model architecture and training strategy that best uses the three types of potentially available data for a non-English translation task: source-pivot, pivot-target, and source-target bilingual datasets.

5.3.1 Multilingual Model: Many-to-Many

A multilingual model is a simple way to use all three types of bilingual data for a pivot-based scenario when a single encoder models all input languages and a single decoder generates all output languages (Figure 5.7). This scheme is called *many-to-many*, and it pursues the ultimate degree of sharing by training a model for various translation directions without the restriction of a common input/output. It is supposed to learn both the multi-speaker property of a one-to-many decoder and the cross-linguality of a many-to-one encoder. Typically, a many-to-many model is trained with bilingual corpora of language pairs involving English in both directions, e.g. German→English and English→German, by flipping its sides. In this thesis, we study a specific case of the pivot-based setup, where the model is trained for source→pivot, pivot→target, and source→target but not the opposite directions.

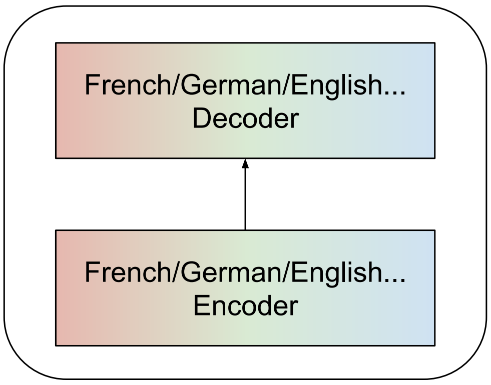

French/German/English... Decoder
French/German/English... Encoder

Figure 5.7: Many-to-many multilingual model (single encoder/decoder).

Unlike the many-to-one case (Section 5.2.1), a many-to-many model's decoder can generate sentences in multiple languages. Nonetheless, it is expected to generate text in only one of the output languages without code-switching, i.e. alternating several languages in the output. To do so, a *language identifier*, e.g. `<2de>` ("to German"), is appended at the beginning of the input sentence or placed as the initial token of the output hypothesis. This provides explicit information of the desired output language to the single shared decoder. For a balanced performance, it is important to divide the decoder capacity effectively over the respective language pairs and regularize both in training.

An interesting by-product of the many-to-many training is zero-shot translation, where the model performs a translation task for which it has never been trained [Johnson & Schuster$^+$ 17]. Let us take an example of how a many-to-many model trained only for French→English and English→German can be used for a French→German task. In this model, the encoder learns to produce the representation of a French input in a shared space with English. The decoder learns to generate a German sentence from English representations, whose space is shared with French representations. With the ideal cross-linguality of the encoder, the decoder should not have a clue of the source language and will translate any encoder representations into the designated target language. Therefore, if a French sentence is given with the language identifier `<2de>`, the model is able to perform French→German translation even if it has not seen such examples in training. As in this example, zero-shot translation is usually employed for non-English language pairs while the model is trained for from-/to-English language pairs.

[Johnson & Schuster$^+$ 17] first propose zero-shot translation with many-to-many models, but the performance is not practical except between very close languages, e.g. Spanish→Portuguese. [Arivazhagan & Bapna$^+$ 19] claim that the poor zero-shot performance is because the model cannot learn to attribute the target language with its corresponding token in the source sentence. To tackle this problem, they add an additional loss function on the encoder to force representational invariance across languages.

5.3.2 Sequential Transfer

In Section 5.2.2, the concept of sequential transfer is introduced as a counterpart of a multilingual model for to-English translation tasks. The same concept can be adapted to a non-English child task with two parents: source→pivot and pivot→target (Figure 5.8):

1. Pre-train a source→pivot model with a source-pivot bilingual corpus and a pivot→target model with a pivot-target bilingual corpus.
2. Initialize the source→target model with the source encoder from the pre-trained source→pivot model and the target decoder from the pre-trained pivot→target model.

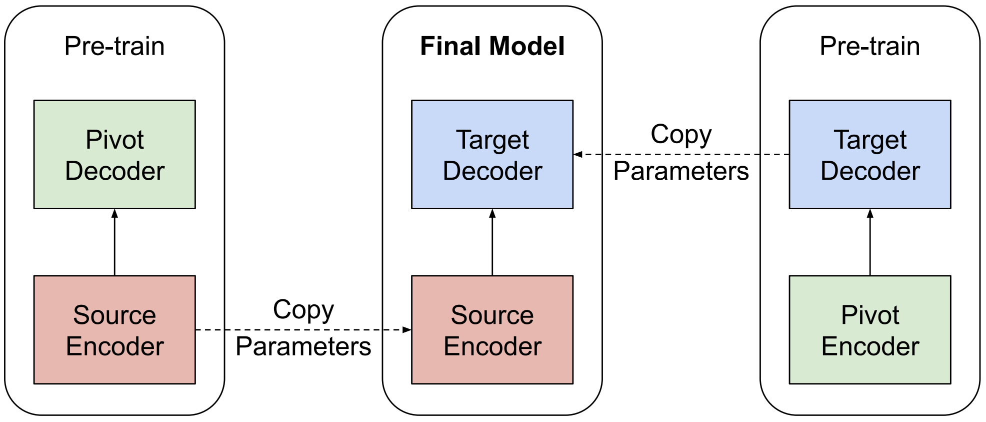


Figure 5.8: Pivot-based sequential transfer.

3. Continue the training with a source-target bilingual corpus.

In the source→pivot training, the encoder is optimized for encoding the source language, and in the pivot→target model training, the decoder is optimized for decoding into the target language. In this way, we can initialize all parameters of the child model, so the pivot language encoder and decoder are discarded after pre-training. If we skip the last step (for zero-resource cases) and perform the source→target translation directly, it corresponds to zero-shot translation.

Thanks to the pivot language, we can pre-train a source encoder and a target decoder without changing the model architecture or training objective for NMT. In contrast to the to-English case, the encoder and the decoder are pre-trained exactly for the source and the target language of the final task, respectively. Thus there is no mismatch of vocabulary or syntax between the parents and the child on each input/output side.

Nonetheless, the main caveat of the sequential transfer in Figure 5.8 is that the source encoder is trained to be used by a pivot decoder, while the target decoder is trained to use the outputs of a pivot encoder — not of a source encoder. The independently pre-trained components should accommodate to each other in the fine-tuning, which is harder when the main task data is not large enough. Also, this discrepancy makes zero-shot translation impossible. Accordingly, we propose three different techniques to mitigate the inconsistency of source→pivot and pivot→target pre-training stages.

Step-wise Pre-Training

A simple remedy to make the pre-trained encoder and decoder refer to each other is to train a single NMT model for source→pivot and pivot→target in consecutive steps (Figure 5.9):

1. Train a source→pivot model with a source-pivot bilingual corpus.
2. Continue the training with a pivot-target bilingual corpus, while freezing the encoder parameters of 1.

In the second step, a target decoder is trained to use the outputs of the pre-trained source encoder as its input. Freezing the pre-trained encoder ensures that, even after the second step, the encoder is still modeling the source language although we are training the NMT model for pivot→target. Without the freezing, the encoder completely adapts to the pivot language input and is likely to forget source language sentences. This way, the data used for each component are the same as in plain transfer (encoder is pre-trained with source-pivot and decoder with pivot-target data), but the connectivity and relationship of the components in pre-training make the transfer more

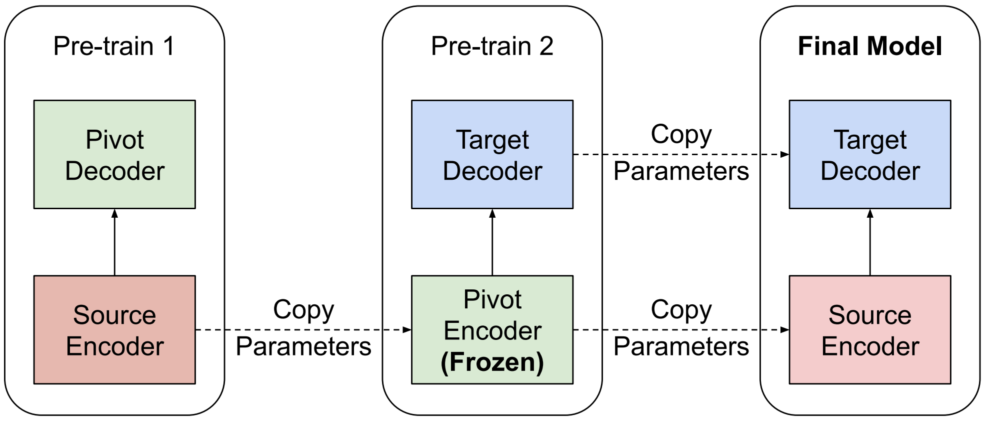


Figure 5.9: Step-wise pre-training for pivot-based transfer.

polished. Also, the decoder pre-training is conditioned on the frozen encoder that outputs the source language; although pre-trained with pivot-target data, the decoder becomes aware of the source sentence structure.

We build a joint vocabulary of the source and pivot languages so that the encoder effectively represents both languages. The frozen encoder is pre-trained for the source language in the first step, but is also able to encode a pivot language sentence in a similar representation space. It is more effective for linguistically similar languages where many tokens are common for both languages in the joint vocabulary.

This step-wise pre-training guides the decoder to take the pre-trained source encoder outputs and produce target language sentences which correspond to the model's behavior in the fine-tuning.

Pivot Adapter

Instead of the step-wise pre-training, we can also postprocess the network to enhance the connection between the source encoder and the target decoder, which are pre-trained individually. The idea is that, after the pre-training steps, we adapt the source encoder outputs to the pivot encoder outputs with which the target decoder is more familiar. The adapter component is trained with source-pivot bilingual data, and the adapted representations of the source language lie in a similar space to the pivot encoder (Figure 5.10). Since the target decoder is pre-trained for pivot→target and accustomed to receive the pivot encoder outputs, it should process the adapted encoder outputs better than the original source encoder outputs (Figure 5.11). This approach is inspired by the cross-lingual mapping of Section 5.2.2. Note that it is essential to fine-tune the full model (encoder, adapter, decoder) with source-target bilingual data to ensure the connection around the adapter. We study such an adapter with different complexities: linear mapping and self-attention layers.

Linear Mapping. The simplest connection between a source encoder and a pivot encoder is a linear transformation with a bias vector. With a small source-pivot bilingual corpus $\mathcal{C}_{f,g}$ in hand, we can learn a linear mapping between the two representation spaces:

1. Encode the source sentences with the source encoder of the pre-trained source→pivot model and the pivot sentences with the pivot encoder of the pre-trained pivot→target model:

$$f_1^J \mapsto \mathbf{h}_{f,1}^J \tag{5.3}$$

$$g_1^K \mapsto \mathbf{h}_{g,1}^J \tag{5.4}$$

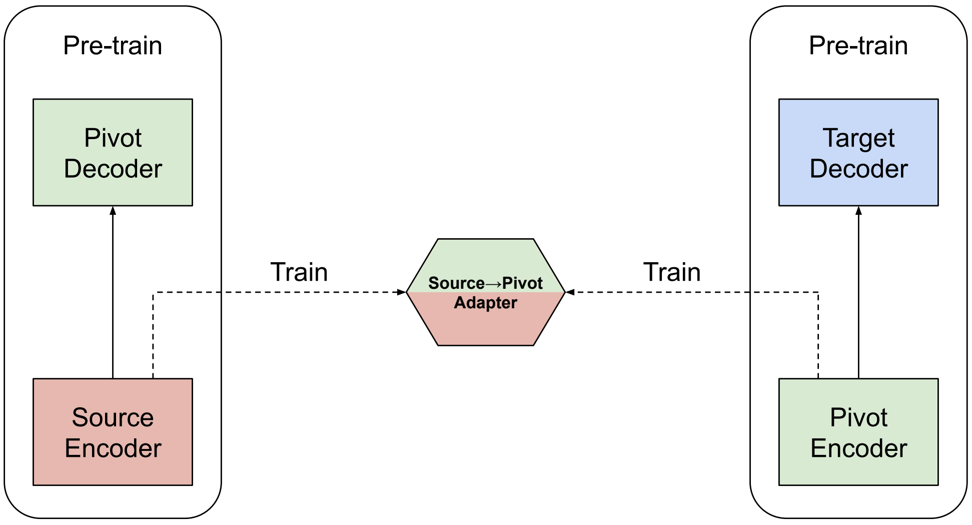


Figure 5.10: Training of a pivot adapter for pivot-based sequential transfer.

2. Apply a pooling to the result of 1, extracting representation vectors for each sentence pair:

$$\mathbf{h}_f = \text{pool}(\mathbf{h}_{f,1}^J) \tag{5.5}$$

$$\mathbf{h}_g = \text{pool}(\mathbf{h}_{g,1}^K) \tag{5.6}$$

3. Train a mapping $\mathbf{W}_{f \to g} \in \mathbb{R}^{D_{\text{hid}} \times D_{\text{hid}}}$ to minimize the distance between the pooled representations $\mathbf{h}_f, \mathbf{h}_g \in \mathbb{R}^{D_{\text{hid}}}$, where the source representation is first fed to the mapping:

$$\mathbf{W}_{f \to g} = \underset{\mathbf{W}}{\text{argmin}} \sum_{(\mathbf{h}_f, \mathbf{h}_g)} \|\mathbf{W} \cdot \mathbf{h}_f - \mathbf{h}_g\|^2 \tag{5.7}$$

Introducing matrix notations $\mathbf{H}_f, \mathbf{H}_g \in \mathbb{R}^{D_{\text{hid}} \times |\mathcal{C}_{f,g}|}$, which concatenate the pooled representations of all n sentences for each side in the source-pivot corpus, Equation 5.7 can be rewritten as:

$$\mathbf{W}_{f \to g} = \underset{\mathbf{W}}{\text{argmin}} \|\mathbf{W} \cdot \mathbf{H}_f - \mathbf{H}_g\|^2 \tag{5.8}$$

which can be easily computed by singular value decomposition (SVD) for a closed-form solution, if we put an orthogonality constraint on $\mathbf{W}$ [Xing & Wang$^+$ 15]. The resulting optimization is also called the Procrustes problem.

Note that we assume the linear mapping to be independent of the position. The pooling at Step 3 first aggregates the representation sequence into one vector, leaving out the positional information and the length of the sequences and simplifying the adapter training. The learned mapping is multiplied identically to encoder outputs of all positions in the final source→target tuning and also in decoding.

Self-Attention Layers. A position-invariant linear mapping may be too crude to model the relationship between sequences of complicatedly encoded representations. As a more sophiscated adapter, sequential encoding layers like self-attention can be employed, which map a sequence of source encoder outputs directly to a sequence of pivot encoder outputs. This considers the sequential interaction between the representation vectors and also introduces non-linearity in the adaptation flow.

Training of a self-attentive adapter can also be done with source-pivot bilingual data. However, a self-attention layer cannot change the sequence length from its input, so special care needs to be

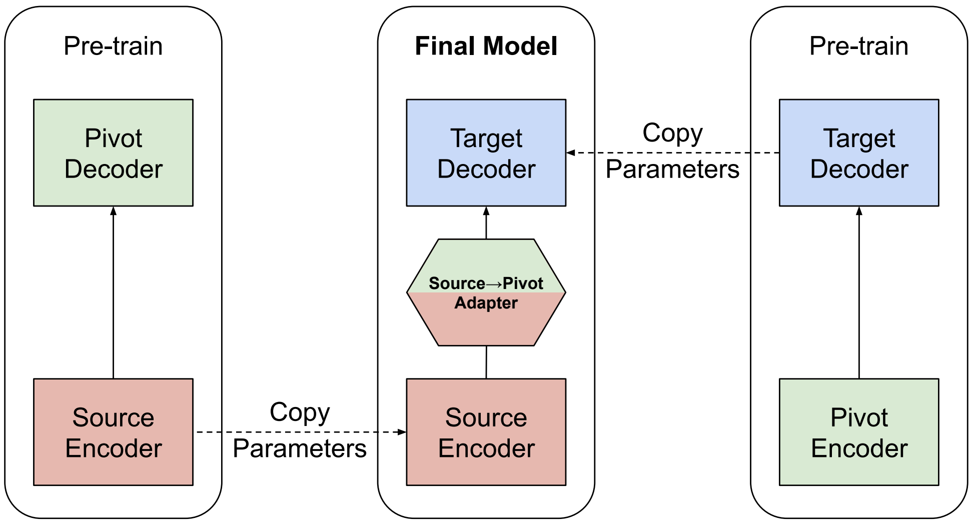


Figure 5.11: Building the final model with pre-trained components and a pivot adapter.

taken to handle the different lengths of a source sentence and a pivot sentence. For this purpose, instead of pooling as in the linear mapping training, we append padding tokens (e.g. `<pad>`) to the shorter sequence. The core steps of training a self-attentive adapter are as follows, assuming the source sequence length is longer than the pivot sequence length ($J > K$):

1. Append padding tokens to the pivot sentences:

$$g_1^K \mapsto g_1^J \tag{5.9}$$

   where $g_j =$ `<pad>` for $j = K+1, ..., J$.

2. Encode the source sentences with the source encoder of the pre-trained source→pivot model and the pivot sentences with the pivot encoder of the pre-trained pivot→target model:

$$f_1^J \mapsto \mathbf{h}_{f,1}^J \tag{5.10}$$

$$g_1^J \mapsto \mathbf{h}_{g,1}^J \tag{5.11}$$

3. Train a self-attentive adapter a_θ to minimize the distance between the representations $\mathbf{h}_{f,j}, \mathbf{h}_{g,j} \in \mathbb{R}^{D_\text{hid}}$ for every position j:

$$\hat{\theta} = \operatorname*{argmin}_{\theta} \sum_{(\mathbf{h}_{f,1}^J, \mathbf{h}_{g,1}^J)} \sum_{j=1}^{J} \|\text{selfatt}_\theta(\mathbf{h}_{f,1}^J)_j - \mathbf{h}_{g,j}\|^2 \tag{5.12}$$

For the training examples whose pivot sentence length is longer than the source sentence, the steps proceed analogously by padding the source sentence at the end. Note that, for an unseen test set after the training, we cannot arbitrarily lengthen source input sentences since the corresponding pivot sentence is unknown; we always use the original source sentence as input in the inference for the final source→target task. For the length difference, we also utilized an explicit source-pivot length model, but there was no clear improvement in the final performance; we decided to keep the padding approach, which is more straightforward.

Despite using an adapter, we still have a gap between the main task training and the pre-trainings, since the transferred components are trained independently of each other. Therefore, after the adapter has been attached, all components must be optimized together with the source→target data.

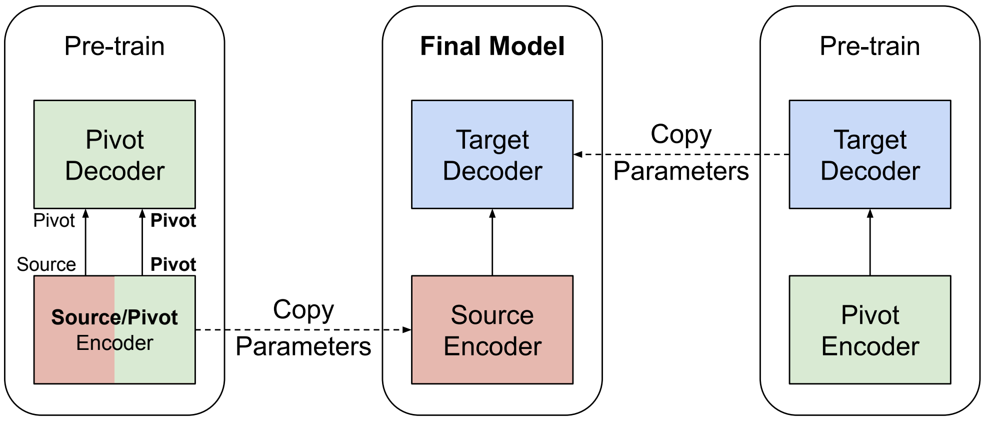


Figure 5.12: Cross-lingual encoder pre-training for pivot-based sequential transfer.

Cross-lingual Encoder

Lastly, we address the discrepancy between the pre-trained encoder and decoder in Figure 5.8 by using an encoder that produces similar representations for bilingual source-pivot sentences. We modify the source→pivot pre-training procedure to force the encoder to have cross-linguality over source and pivot languages, modeling source and pivot sentences in the same mathematical space. In source-pivot pre-training, we add an additional training objective of translating the pivot sentence to the same pivot sentence in order to achieve the cross-lingual property of the encoder. This cross-lingual space facilitates smoother learning of the final source→target model, because the decoder is pre-trained to translate the pivot language. In the encoder pre-training step, the encoder is fed with sentences of both source and pivot languages, which are processed by a shared decoder that outputs only the pivot language (Figure 5.12). In this way, the encoder learns to produce representations in a shared space regardless of the input language, since they are used in the same decoder.

However, the same input/output in autoencoding encourages merely copying the input; it is said to be not proper for learning the complex structure of the data domain [Vincent & Larochelle$^+$ 08]. Denoising autoencoder addresses this by corrupting the input sentences with artificial noise [Hill & Cho$^+$ 16]. Learning to reconstruct clean sentences, it encodes the linguistic structures of natural language sentences, e.g. word order, better than copying. Here are the noise types we use [Edunov & Ott$^+$ 18]:

- Drop tokens randomly with a probability p_d
- Replace tokens with a `<BLANK>` token randomly with a probability p_r
- Permute the token positions randomly so that the difference between an original index and its new index is less than or equal to d_p

The cross-lingual encoder is expected to work best in combination with step-wise pre-training. In the second step of the original step-wise pre-training (Figure 5.9), pivot sentences are encoded with the source encoder and sent as an input to the decoder. The decoder effectively receives random semantics although they are mathematically in the same space as the source representations. This could confuse the training for the final task where only sensible semantics in the source language are given to the decoder. The cross-lingual encoder directly addresses this problem by learning similar representations for similar source and pivot inputs. When the decoder is trained to process the (fixed) cross-lingual encoder with pivot-target bilingual data, the pivot sentences are encoded to a similar representation to a comparable source sentence. Ideally, it also learns to translate source sentences because of the cross-lingual encoder even with the pivot-target bilingual data.

The frozen cross-lingual encoder in the second step of the step-wise procedure reduces the gap between pre-training and the final task training.

The key idea of all three methods is to build a closer connection between the pre-trained encoder and decoder via a pivot language. The difference is in when we do this job: The cross-lingual encoder changes the encoder pre-training stage (source→pivot), while step-wise pre-training modifies the decoder pre-training stage (pivot→target). The pivot adapter is applied after all pre-training steps. Note that these techniques are not exclusive and some of them can complement others for better performance of the final model, as we explained in the cross-lingual encoder in step-wise pre-training.

Experiments

We evaluate the proposed sequential transfer techniques in two non-English WMT 2019 news translation tasks: German→Czech (Section B.8) and French→German (Section B.9). The former has a common low-resource condition (Section 1.2) for non-English language pairs. The latter comes with a considerable amount of bilingual data (around 2.5M sentence pairs), which is the case between several non-English languages whose roots are geographically close to each other. Both tasks have much abundant bilingual data in source-pivot and pivot-target with English as the pivot language. Detailed corpus statistics of those pivot-based parent tasks are given in Section B.2, B.10, and B.11. Note that pre-training for German→English also used the back-translated data from [Sennrich & Haddow^{+} 16a] (around 4.2M sentence pairs). For the experiments in this section, sentence pairs in the training data were filtered out if the length of either side was more than twice as long as the other or the amount of alphabet-only tokens was smaller than 65% of all tokens in the sentence.

For all sequential transfer setups, we learned BPE vocabularies for each language individually with 32k merge operations, except for cross-lingual encoder training with joint BPE only over source and pivot languages. This is for the modularity of the pre-trained models: for example, a French→English model trained with joint French/English/German BPE could be transferred smoothly to a French→German model, but would not be optimal for a transfer to e.g. a French→ Korean model. Once we have pre-trained an NMT model with separate BPE vocabularies, we can reuse it for various final language pairs without wasting the unused portion of the subword vocabularies (e.g. German-specific tokens in building a French→Korean model). The NMT model training and transfer were done with the OpenNMT toolkit [Klein & Kim^{+} 17], where we used an initial learning rate of 0.0001 and a checkpoint frequency of 10k updates. For the low-resource fine-tuning (German→Czech), we performed checkpointing at every epoch. The pivot adapter was trained using the Muse toolkit [Conneau & Lample^{+} 18], which was originally developed for bilingual word embeddings but we adjusted it for matching sentence representations.

As a comparative baseline, we build a direct model using only the source→target training data. In Section 4.3, we have already seen that synthetic bilingual data is highly effective in enhancing a plain baseline model. Exploiting pivot-based parent tasks, the direct baseline is augmented with pivot-based back-translation [Bertoldi & Barbaiani^{+} 08], where the entire available pivot-target data is translated into the source language. In addition, we train a many-to-many multilingual model for the tasks whose input or output language matches the final task at least, i.e. source→pivot, pivot→target, and source→target but not the opposite directions. Furthermore, we also show results of the conventional pivoting. Except for pivoting, all other baselines are built with BPE codes jointly learned over all input/output languages with 32k merges.

Effect of Sequential Transfer Methods. In Table 5.6, we report the principal results of sequential transfer methods with fine-tuning using given source-target bilingual data. As for baselines, pivoting is already very strong in both tasks. The direct source→target model is

Table 5.6: Comparison of sequential transfer schemes for non-English translation tasks. The fine-tuning was done with real bilingual data of source→target.

(a) WMT 2019 German→Czech

Data Size (#sents)			Method	newstest2012		newstest2013	
				Bleu	Ter	Bleu	Ter
de-cs	de-en	en-cs		[%]	[%]	[%]	[%]
-	9.1M	49M	Pivoting	18.0	73.6	21.3	68.8
226k	-	-	Direct baseline	12.0	79.7	13.5	76.3
226k	9.1M	49M	+ Pivot-based back-translation	15.7	76.5	18.5	72.0
			Multilingual	14.9	76.6	16.5	73.2
226k	9.1M	49M	Plain transfer	15.4	75.4	18.0	70.9
			+ Pivot adapter	15.9	75.0	18.7	70.3
			+ Cross-lingual encoder	15.0	75.9	17.6	71.4
			+ Pivot adapter	15.6	75.3	18.1	70.8
			Step-wise pre-training	15.6	75.0	18.1	70.9
			+ Cross-lingual encoder	**16.2**	**74.6**	**19.1**	**69.9**

(b) WMT 2019 French→German

Data Size (#sents)			Method	newstest2012		newstest2013	
				Bleu	Ter	Bleu	Ter
fr-de	fr-en	en-de		[%]	[%]	[%]	[%]
-	35M	9.1M	Pivoting	20.6	68.9	22.3	68.5
2.5M	-	-	Direct baseline	20.1	69.8	21.9	69.2
2.5M	35M	9.1M	+ Pivot-based back-translation	21.1	68.2	22.6	68.1
			Multilingual	19.9	69.4	21.7	69.2
2.5M	35M	9.1M	Plain transfer	20.8	69.2	22.7	68.3
			+ Pivot adapter	20.7	69.2	22.9	67.9
			+ Cross-lingual encoder	**21.0**	**69.0**	22.8	68.0
			+ Pivot adapter	20.9	69.1	22.8	68.2
			Step-wise pre-training	20.6	69.4	22.7	68.3
			+ Cross-lingual encoder	20.9	69.4	**23.1**	**68.0**

weak but considerably improved by pivot-based synthetic data, especially in the low-resource German→Czech task. Multilingual models cannot outperform pivoting and the improved direct baselines in both tasks.

Plain transfer of pre-trained encoder/decoder without additional techniques shows a nice improvement over the direct baseline: up to +4.5% Bleu for German→Czech and +0.8% Bleu for French→German. The pivot adapter provides an additional boost of a maximum +0.7% Bleu. Cross-lingual encoder pre-training proved to be not significantly effective in the plain transfer setup. It shows 0.4% Bleu worse performance in German→Czech and only minor improvements over plain transfer in French→German. We conjecture that the cross-lingual encoder needs a lot more data to be fine-tuned for another decoder, where the encoder capacity is basically divided into two languages at the beginning of the fine-tuning. On the other hand, the pivot adapter directly improves the connection to an individually pre-trained decoder, which works nicely with the small amount of fine-tuning data.

The pivot adapter gives an additional improvement on top of the cross-lingual encoder; up to +0.6% Bleu in German→Czech. In this case, we extract source and pivot sentence representations from the same shared encoder for training the adapter. The encoder shares all of its parameters over the two languages, which offers a strong source-pivot connection, but its mathematical space is completely different from what a pre-trained decoder expects. According to the French→German results, we see that the final model can be comparable to the plain transfer once we fine-tune the encoder and decoder together with 2.3M source-target sentence pairs. With ten times smaller source-target data (German→Czech), it is more difficult to optimize the performance for source→target, where the encoder capacity is basically divided into two languages at the beginning of the fine-tuning. On the other hand, the pivot adapter directly improves the connection to an individually pre-trained decoder, which works nicely in this low-resource scenario.

Step-wise pre-training alone gives no improvement against plain transfer, but shows the best performance in both tasks when combined with the cross-lingual encoder: up to +5.6% Bleu in German→Czech and +1.2% Bleu in French→German, compared to the direct baseline. Step-wise pre-training prevents the cross-lingual encoder from degeneration, since the pivot→target pre-training also learns the encoder-decoder connection with a large amount of data — in addition to the source→target tuning step afterwards. Note that the pivot adapter, which inserts an extra layer between the encoder and decoder, is not appropriate after the step-wise pre-training; the decoder is already trained to correlate well with the pre-trained encoder. We experimented with the pivot adapter on top of step-wise pre-trained models — with or without cross-lingual encoder — but obtained detrimental results.

In German→Czech, our best sequential transfer results outperform all comparative baselines except pivoting. In French→German, sequential transfer is comparable to the best baseline; slightly better in `newstest2013` but worse in `newstest2012`. This shows that, if the source-target data is not sufficient, pivoting is still a viable option which is hard to reach by sequential transfer with the small fine-tuning data. When we have a relatively large amount of data for source→target fine-tuning, sequential transfer is less impactful against the direct baseline but its final performance overcomes pivoting. Note that the additional techniques for sequential transfer have a weaker impact in this high-resource scenario. These techniques are developed under the assumptions of low-/zero-resource setups, where it is harder to adapt the pre-trained components to each other. We think that 2.3M sentence pairs are enough for the component adjustments, which is why the adapter or cross-lingual encoder makes less of a contribution.

Data for Fine-Tuning. We have observed in Table 5.6 that comparison among the pivot-based NMT schemes yields slightly different conclusions for the two translation tasks, one of which is low-resource and the other high-resource in the final source→target task. In order to investigate further the impact of the amount of the final task data, we present the results of the best sequential transfer setting (step-wise pre-training with cross-lingual encoder) in different fine-tuning conditions: 1) real bilingual data, 2) synthetic bilingual data, 3) combination of real and synthetic bilingual data, and 4) no fine-tuning at all.

The second condition represents zero-resource scenarios where no source-target data is at hand. Here, synthetic bilingual corpora generated from source-pivot and/or pivot-target data are the only training signals that can be used in the fine-tuning stage. In Table 5.7, we performed sequential transfer with the same pivot-based back-translation data that augmented the direct baseline. Using the synthetic data outperforms the real data in fine-tuning for the German→Czech task. The synthetic data is more than 200 times larger than the real one, which improves the sequential transfer by up to +1.6% Bleu. On the other hand, the synthetic data underperforms the real data in the French→German task fine-tuning; the real data is already sufficient and the synthetic data is only 3.6 times larger than that.

Combining synthetic and real bilingual data in fine-tuning, our sequential transfer results are comparable to pivoting in the German→Czech task, where Ter scores are even lower. In the French→German task, the combination gives an additional improvement up to +1.0% Bleu and -1.8% Ter against using only the real data, which is now clearly superior to all of the baseline

Table 5.7: Comparison of fine-tuning data for sequential transfer in non-English translation tasks. The transfer scheme is step-wise pre-training with cross-lingual encoder. The synthetic data was generated by pivot-based back-translation.

(a) WMT 2019 German→Czech

Data Size (#sents)					newstest2012		newstest2013	
					Bleu	Ter	Bleu	Ter
de-cs	de-en	en-cs	Method	Fine-Tuning Data	[%]	[%]	[%]	[%]
-	9.1M	49M	Pivoting	-	18.0	73.6	21.3	68.8
226k	-	-	Direct baseline		12.0	79.7	13.5	76.3
226k	9.1M	49M	+ Synthetic data		15.7	76.5	18.5	72.0
			Multilingual		14.9	76.6	16.5	73.2
226k	9.1M	49M	Sequential transfer	Real	16.2	74.6	19.1	69.9
				Synthetic	17.6	73.6	20.7	68.7
				Real + Synthetic	**18.0**	**72.7**	**21.3**	**68.0**
				None	14.1	76.8	16.5	73.5

(b) WMT 2019 French→German

Data Size (#sents)					newstest2012		newstest2013	
					Bleu	Ter	Bleu	Ter
fr-de	fr-en	en-de	Method	Fine-Tuning Data	[%]	[%]	[%]	[%]
-	35M	9.1M	Pivoting	-	20.6	68.9	22.3	68.5
2.5M	-	-	Direct baseline		20.1	69.8	21.9	69.2
2.5M	35M	9.1M	+ Synthetic data		21.1	68.2	22.6	68.1
			Multilingual		19.9	69.4	21.7	69.2
2.5M	35M	9.1M	Sequential transfer	Real	20.9	69.4	23.1	68.0
				Synthetic	20.4	68.2	21.8	68.3
				Real + Synthetic	**21.9**	**67.6**	**23.4**	**67.4**
				None	17.3	72.1	18.0	72.7

results. Note that the real source-target bilingual data were oversampled to make the ratio of real and synthetic data 1:2. For the best performance, we argue that it is essential to include synthetic bilingual data in the sequential transfer fine-tuning. Pivot-based synthetic data can be generated from the parent bilingual corpora without an extra data source, e.g. monolingual data. Such synthetic data is normally of high quality since it is generated from strong pivot-based parent models.

Without any fine-tuning (the last row of Table 5.7a and 5.7b), the performance is naturally worse than doing fine-tuning but the model is still able to produce meaningful translations. It shows even better results than the direct baseline in the German→Czech task without seeing any source→target translations in training. Cross-lingual encoder helps the smooth connection between two parent datasets and, after step-wise pre-training, the decoder can use the encoder representations directly; the proposed techniques enable zero-shot translations to a reasonable extent.

Zero-Shot Scenario. We have previously seen that zero-shot translation is indeed possible with the proposed sequential transfer techniques. A zero-shot scenario assumes that the model is not trained with any real or synthetic bilingual data from source to target, which is not realistic in terms of practical system building for the best performance. However, it offers a rawer measure of the cohesion between the pre-trained model components and is itself an interesting testbench

Table 5.8: Zero-shot translation results in non-English translation tasks.

(a) WMT 2019 German→Czech

Data Size (#sents)				`newstest2012`		`newstest2013`	
				Bleu	Ter	Bleu	Ter
de-cs	de-en	en-cs	Method	[%]	[%]	[%]	[%]
-	9.1M	49M	Pivoting	18.0	73.6	21.3	68.8
			Multilingual	5.9	101.9	6.3	99.8
-	9.1M	49M	Plain transfer	0.1	-	0.1	-
			+ Pivot adapter	0.1	-	0.1	-
			Step-wise pre-training	6.0	92.1	6.5	87.8
			+ Cross-lingual encoder	**14.1**	**76.8**	**16.5**	**73.5**

(b) WMT 2019 French→German

Data Size (#sents)				`newstest2012`		`newstest2013`	
				Bleu	Ter	Bleu	Ter
fr-de	fr-en	en-de	Method	[%]	[%]	[%]	[%]
-	35M	9.1M	Pivoting	20.6	68.9	22.3	68.5
			Multilingual	14.1	79.1	14.6	79.1
-	35M	9.1M	Plain transfer	0.1	-	0.2	-
			+ Pivot adapter	0.1	-	0.1	-
			Step-wise pre-training	11.0	81.6	11.5	82.5
			+ Cross-lingual encoder	**17.3**	**72.1**	**18.0**	**72.7**

of the cross-lingual capability of neural network architectures.

On this account, we study zero-shot scenarios further in Table 5.8. Note that, unlike Table 5.6 and 5.7, the multilingual baselines exclude source→target and target→source directions. First of all, plain transfer, where the encoder and the decoder are pre-trained separately, is poor in zero-shot scenarios. It simply fails to connect different representation spaces of the pre-trained encoder and decoder. In our experiments, neither pivot adapter nor cross-lingual encoder could enhance the zero-shot translation of plain transfer.

Step-wise pre-training solves this problem by changing the decoder pre-training to familiarize itself with representations from an already pre-trained encoder. It achieves zero-shot performance of 11.5% Bleu in French→German and 6.5% Bleu in German→Czech (`newstest2013`), while showing comparable or better fine-tuned performance against plain transfer (see also Table 5.6). With the pre-trained cross-lingual encoder, the zero-shot performance of step-wise pre-training is superior to that of pivot translation in French→German with only a single model. It is worse than pivot translation in German→Czech. We think that the data size of the pivot-target is critical in pivot translation; relatively huge data for English→Czech make the pivot translation stronger. Note again that, nevertheless, pivoting (second row) is very poor in efficiency since it performs decoding twice with the individual models.

We also evaluate our best German→Czech zero-resource model on `newstest2019` and compare it with the participants of the WMT 2019 unsupervised news translation task. Ours yield 17.2% Bleu, which is much better than the best single unsupervised system of the winner of the task (15.5%) [Marie & Sun$^+$ 19]. We argue that, if one has enough source-English and English-target bilingual data for a non-English language pair, it is preferable to adopt pivot-based transfer learning than unsupervised MT — even if there is no source-target bilingual data. In this case, unsupervised MT unnecessarily restricts the data condition to using only monolingual data and its high computational cost does not pay off; simple pivot-based pre-training steps are more efficient and effective.

Pivot Adapter: Variants. As a detailed study, we compare variants of the pivot adapter in Table 5.9. "None" in the "Adapter Training" column shows that a randomly initialized linear layer guides the pre-trained encoder/decoder to harmonize with each other. Of course, when we train the adapter to map source encoder outputs to pivot encoder outputs, the performance gets better. For compressing encoder outputs over positions, average-pooling is better than max-pooling. Against our expectation, self-attention layers provide no better results than linear mapping. They improve over the plain transfer only slightly with a random initialization.

We hypothesize that a self-attentive adapter is too complex for connecting the pre-trained encoder and decoder, increasing the overfitting for the low-resource fine-tuning. On the other hand, a linear adapter plays a regularizer role because all encoder outputs are mapped with the same matrix, which can reduce the overfitting against other sophisticated components in the network.

Table 5.9: Comparison of architecture and training scheme for pivot adapter (German→Czech).

			`newstest2012`		`newstest2013`	
Method	Adapter Architecture	Adapter Training	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
Plain transfer	-	-	15.4	75.4	18.0	70.9
+ Pivot adapter	Linear	None	15.5	75.2	18.2	70.7
		Max-pooled	15.9	74.9	18.4	70.5
		Average-pooled	**15.9**	**75.0**	**18.7**	**70.3**
	Self-attention	None	15.7	74.8	18.3	70.3
		Sequence	15.4	75.1	17.8	70.9

Cross-lingual Encoder: Data for Autoencoding. Table 5.10 empirically verifies that noisy input in autoencoding is beneficial to our cross-lingual encoder. It improves the final translation performance by maximum +2.1% Bleu, compared to using the copying autoencoding objective. As the training data for autoencoding, we also compare purely monolingual data and the pivot side of the source-pivot bilingual data. With the latter, one can expect a stronger signal for a joint encoder representation space, since two different inputs (in source/pivot languages) are used to produce exactly the same output sentence (in pivot language). The results also show that there are slight but consistent improvements when using the pivot part of the bilingual data.

Table 5.10: Comparison of autoencoding training data for cross-lingual encoder (French→German). All results are in the zero-shot setting.

			`newstest2012`		`newstest2013`	
Method	Pivot Language Data for Autoencoding	Noisy Input	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
Step-wise pre-training	-	-	11.0	81.6	11.5	82.5
+ Cross-lingual encoder	Monolingual	No	15.3	76.6	15.7	77.7
		Yes	16.9	73.2	17.5	73.6
	Source-Pivot	No	15.2	76.9	15.9	77.3
		Yes	**17.3**	**72.1**	**18.0**	**72.7**

5.3.3 Cascaded Architecture (Pivoting)

Until now, we have covered the approaches that aim to build a single encoder-decoder model for the end task (source→target) of a pivot-based scenario. These approaches avoid the conventional cascading of source→pivot and pivot→target models in the inference, halving the decoding time. The sequential transfer scheme achieves a comparable or better performance than the cascaded approach (pivoting), while translating with a single model. Meanwhile, we also observed that pivoting is a very solid baseline that cannot be surpassed easily despite its inefficiency. In this section, we attempt to reform the cascaded architecture of pivoting, leaving the single-model requirement behind and concentrating only on the translation performance.

As stressed earlier several times, pivoting (Figure 5.13) propagates the errors of the first model (source→pivot) to the second model (pivot→target) without any revision, possibly amplified even further during the second hypothesis search. In the following, we explore how to train the cascaded architecture on source-to-target bilingual data, which might optimize the output of the first model to be better suited to the second model, reducing critical errors inbetween. In order to do that, we suggest new interfaces through which both models are merged together, potentially improving the final translation performance. Note that for all concepts in this chapter, we presuppose two pre-trained models for source→pivot and pivot→target, which will both be Transformer models.

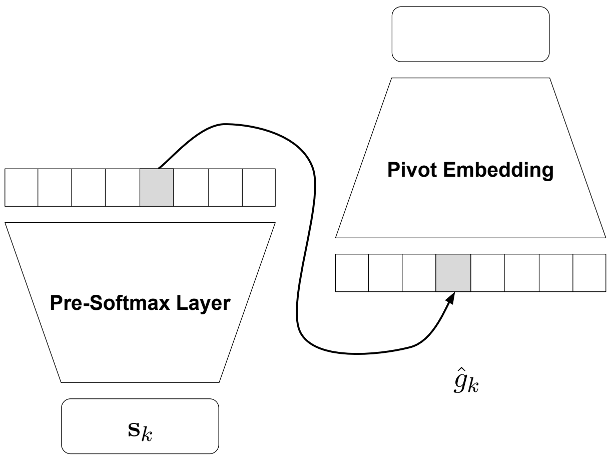


Figure 5.13: Passing single-best (one-hot) hypothesis for vanilla pivoting.

Interface: Weighted Sum of Embeddings

In pivoting, the hypothesis in the pivot language is passed to the pivot→target model. What is not passed is the grounds for the decisions, i.e. how the source→pivot model ended up with the hypothesis. For example, there are uncertainties when deciding between two different tokens which are equally likely. Also, passing discrete decisions of the hypothesis means a nondifferentiable node between the first and the second models, which does not allow an integrated training throughout the whole network, i.e. from target to source via pivot language.

To counteract this, instead of the best hypothesis, we pass the softmax output vectors of the source→pivot model to the pivot→target model (Figure 5.14). The softmax probability measures the confidence of the first model in deciding the source→pivot translations. This delivers richer information to the second model, retaining all alternative decisions that the first model might have taken. The second model has a choice to actively make use of the alternative hypotheses or ignore them. If the best hypothesis of the first model has an error, this concept gives a chance to the alternatives which might be correct.

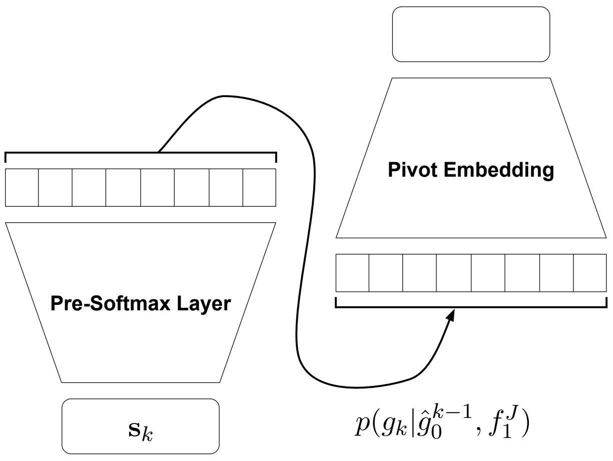


Figure 5.14: Passing posterior probabilities for integrated pivoting.

For a tighter integration, we let the embedding of the first decoder and the embedding of the second encoder share the same weights. For this, we first train the source→pivot model and extract its output embedding, with which we initialize the input embedding of the pivot→target model. Then we train the second model with the initialized input embedding fixed. At inference time, the softmax vector is multiplied by the shared pivot embedding, effectively feeding the second model with the weighted sum of the pivot embeddings.

Most importantly, the pre-trained models can be concatenated through the softmax probabilities and further trained all together using a source-target bilingual corpus. The softmax layer inbetween is differentiable once the left-to-right decoding decisions of the first model have been given. Thus we first generate a pivot sentence from the source side of the corpus to make source-pivot-target trilingual data and train the concatenated model with it. Note that, in this integrated training, we update all vectors of the pivot embedding in one step instead of only the position of the one-hot coded token. The gradient of each embedding vector is scaled in magnitude with the corresponding softmax probability. We also call this post-procedure fine-tuning for the cascaded architecture.

During pre-training, the embedding layers are trained to expect the one-hot vector as input, and are not familiar with posterior probabilities. In order not to confuse the model too much, we insert a new modification layer right after the first softmax layer. It takes every entry of a probability distribution $\mathbf{p} = [p_1, ..., p_v, ... p_V] \in \mathbb{R}^V$ to the power of $\gamma \in [0, 1]$ and renormalizes it:

$$\bar{p}_v = \frac{p_v^\gamma}{\sum_{v'=1}^{V} p_{v'}^\gamma} \tag{5.13}$$

A larger γ makes the distribution more peaked by amplifying the probability of the best hypothesis word while squeezing that of the alternative words. The output distribution thus becomes closer to a one-hot vector that the pre-trained second model anticipates. On the other hand, a smaller γ flattens the distribution and reduces the influence of the best hypothesis. Figure 5.15 illustrates how this layer modifies a probability distribution. The input distribution has the highest probability for the third word and the lowest for the sixth word. This trend does not change but the highest one gets even higher after the layer with $\gamma = 4$, whilst the least likely one becomes almost zero. In this case, the two most probable predictions take up roughly 90% of the probability mass, making the decision more certain and familiar to the second model. In practice, a very large γ might cause all entries of the vector, including the highest-scoring one, to be rounded to zero before renormalization; for numerical stability, we also divide the softmax output vector by its maximum before feeding it to the modification layer.

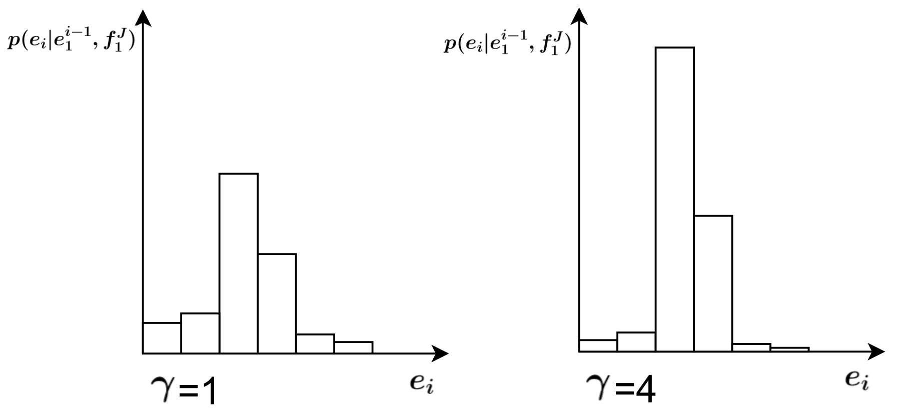


Figure 5.15: Exemplary case for sharper posterior applied to a probability distribution.

Interface: Decoder State

The second interface we propose is to skip the softmax layer for the pivot hypothesis and just feed the decoder state of the first model to the second encoder (Figure 5.16). This is based on the assumption that the output layer is only necessary for making a discrete decision possible and does not yield any important additional information. This means that the last state of the decoder already has all the required information of the translation process. Our final purpose is not to produce a good pivot hypothesis in the middle but to output target translations. For the second model to produce a good target output, it may not need lexical decisions in the pivot language but state vectors with the necessary information may suffice. Note that, even though the softmax output is not passed to the second model, we still need to process the softmax layers of the first model in order to predict the next tokens in the pivot language and get a state vector sequence of a reasonable length.

The decoder states are already in the dimension of D_{hid} and do not need the embedding mapping for entry to the second model. They are supposed to be passed directly to the first self-attention layer of the second model's encoder. In the case of the Transformer, positional encoding is crucial for a sequence to be used as an input, so it is added to the state vectors before the self-attention begins. Fine-tuning is even more important for the decoder state interface, since the encoder of the second model is never trained to process state vectors. Like the preceding interface, we generate

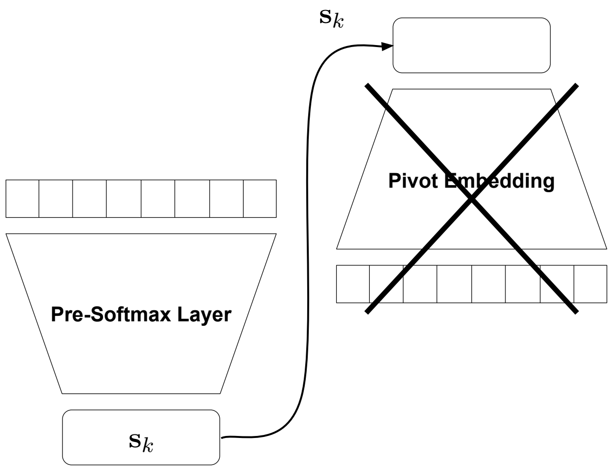


Figure 5.16: Passing decoder state for integrated pivoting.

the pivot part from the source-target training data and use it in the fine-tuning.

Experiments

The cascaded architecture with the new connecting interfaces were evaluated on the same German→Czech and French→German tasks as the experiments of Section 5.3.2 (see corpora statistics in Section B.8-B.11). The baselines were also the same as Section 5.3.2: direct source→ target (improved with synthetic data) and multilingual model. As explained above, we pre-trained the source→pivot model first and then the pivot→target model with a shared embedding matrix for the pivot language. The source→target fine-tuning was done with a ten times lower learning rate (0.00001). Unless otherwise stated, we set $\gamma = 1.0$, which introduces no modification to the source→pivot softmax outputs.

Effect of New Interfaces and Integrated Fine-Tuning. Table 5.11 shows the translation results of cascaded systems with the newly introduced interfaces. The one-hot interface in the cascaded method indicates the conventional pivoting, which outperforms other baselines in German→Czech and is close to the best baseline (direct source→target augmented with a synthetic data) in French→German. Without integrated fine-tuning, the proposed interfaces worsen the performance since the pre-trained pivot→target model did not experience the new types of inputs. The weighted sum of embeddings maintains the performance on a similar level (-0.8% Bleu

Table 5.11: Comparison of interfaces between source→pivot and pivot→target models for pivoting.

(a) WMT 2019 German→Czech

Data Size (#sents)					`newstest2012`		`newstest2013`	
					Bleu	Ter	Bleu	Ter
de-cs	de-en	en-cs	Method	Interface	[%]	[%]	[%]	[%]
226k	-	-	Baseline	-	12.0	79.7	13.5	76.3
226k	9.1M	49M	+ Synthetic data		15.7	76.5	18.5	72.0
			Multilingual		14.9	76.6	16.5	73.2
-	9.1M	49M	Pivoting (Cascaded)	One-hot	**18.0**	**73.6**	**21.3**	**68.8**
				Posterior	17.5	74.4	20.5	69.6
				Decoder State	0.1	-	0.1	-
226k	9.1M	49M	+ Fine-Tuning	Posterior	17.4	73.9	20.7	69.1
				Decoder State	16.8	73.9	19.5	69.7

(b) WMT 2019 French→German

Data Size (#sents)					`newstest2012`		`newstest2013`	
					Bleu	Ter	Bleu	Ter
fr-de	fr-en	en-de	Method	Interface	[%]	[%]	[%]	[%]
2.5M	-	-	Baseline	-	20.1	69.8	21.9	69.2
2.5M	35M	9.1M	+ Synthetic data		21.1	68.2	22.6	68.1
			Multilingual		19.9	69.4	21.7	69.2
-	35M	9.1M	Pivoting (Cascaded)	One-hot	20.6	68.9	22.3	68.5
				Posterior	20.5	69.0	21.9	68.8
				Decoder State	0.1	-	0.1	-
2.5M	35M	9.1M	+ Fine-Tuning	Posterior	**21.1**	**68.0**	**22.8**	**67.5**
				Decoder State	21.2	68.4	22.9	68.0

at most), since it still delivers embedding vectors, though transformed, in the same mathematical space that the pivot encoder is trained to see. The decoder state lies in a different space than the pivot embedding, where the interface does not function at all.

Once we fine-tuned the concatenated model on the bilingual data, the new interfaces bring improvements up to +0.6% Bleu and -1.0% Ter over the one-hot interface in the French→German task. To the best of our knowledge, this is the first time in the literature that the conventional pivoting has made a notable improvement through a structured change in its model architecture and training. The fine-tuning gives a chance for the model to adapt to the new interface and learn from the additional training data of the end task (source→target), which the conventional pivoting has never been able to exploit. By passing the decoder states, we obtain the same improvement as the weighted sum of embeddings. This means our assumption about the output layer is valid; it can be skipped when our end task does not actually require a hypothesis from it.

In the German→Czech task, the fine-tuning does improve with the new interfaces but the final results do not outperform the conventional pivoting. We conjecture that the 226k sentence pairs of the source-target is not sufficient for the interfaces to manifest their effectiveness.

Data for Fine-Tuning. Similarly to the sequential transfer case (Section 5.3.2), we experimented with synthetic bilingual data in fine-tuning the cascaded architectures in Table 5.12. We used the same pivot-based back-translation data and oversampling principle as in Table 5.7. In both tasks, the synthetic data does not improve the previous fine-tuning results in pivoting, either alone or in combination with the real source-target data. The reason could be that the pivot→target model has already seen the pivot and target sides of the synthetic data in pre-training and converged with it, so there is only a marginal effect by continuing to train on it again.

Table 5.12: Comparison of fine-tuning data for pivoting with posterior interface.

(a) WMT 2019 German→Czech

Data Size (#sents)			Method	Fine-Tuning Data	`newstest2012`		`newstest2013`	
					Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
de-cs	de-en	en-cs						
226k	-	-	Baseline		12.0	79.7	13.5	76.3
226k	9.1M	49M	+ Synthetic data		15.7	76.5	18.5	72.0
			Multilingual		14.9	76.6	16.5	73.2
-	9.1M	49M	Pivoting (Cascaded)	-	**18.0**	**73.6**	**21.3**	**68.8**
226k	9.1M	49M	+ Fine-tuning	Real	17.4	73.9	20.7	69.1
				Real + Synthetic	17.8	73.9	20.6	69.4
				Synthetic	17.7	73.9	20.7	69.3

(b) WMT 2019 French→German

Data Size (#sents)			Method	Fine-Tuning Data	`newstest2012`		`newstest2013`	
					Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
fr-de	fr-en	en-de						
2.5M	-	-	Baseline		20.1	69.8	21.9	69.2
2.5M	35M	9.1M	+ Synthetic data		21.1	68.2	22.6	68.1
			Multilingual		19.9	69.4	21.7	69.2
-	35M	9.1M	Pivoting (Cascaded)	-	20.6	68.9	22.3	68.5
2.5M	35M	9.1M	+ Fine-Tuning	Real	**21.1**	**68.0**	**22.8**	**67.5**
				Real + Synthetic	20.7	68.2	22.4	67.8
				Synthetic	20.7	68.2	22.2	67.9

The source and pivot sides of the synthetic data are still fresh to the pre-trained source→pivot model, but as a full chain of training signal from target to source, it seems less powerful than the real data. Note that, here, the pivot sentences are also used in the middle to force the forwarding pass of the source→pivot component and eliminate discrete decision nodes in the gradient path of fine-tuning. In contrast, in the sequential transfer fine-tuning, a single encoder-decoder model learns to construct a direct relation from a synthesized source to a target sentence, which is clearly different from the pre-training signals and proves to be instructive (Table 5.7).

Progress of Fine-Tuning. To understand how the fine-tuning operates in a cascaded architecture, we traced the source→target and source→pivot loss values during a successful fine-tuning run for the French→German task. Note that the source→pivot loss was recorded only for the analysis and did not affect the training. While the source→target loss went down over the epochs, the source→pivot loss increased steadily. This means that, in order to improve the translations for

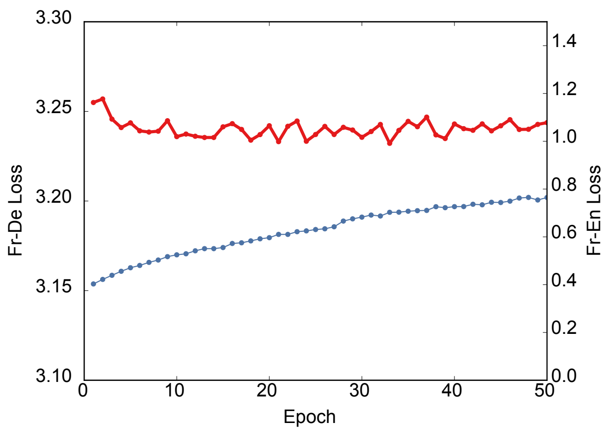


Figure 5.17: Progress of loss values on the development set during fine-tuning (WMT 2019 French→German).

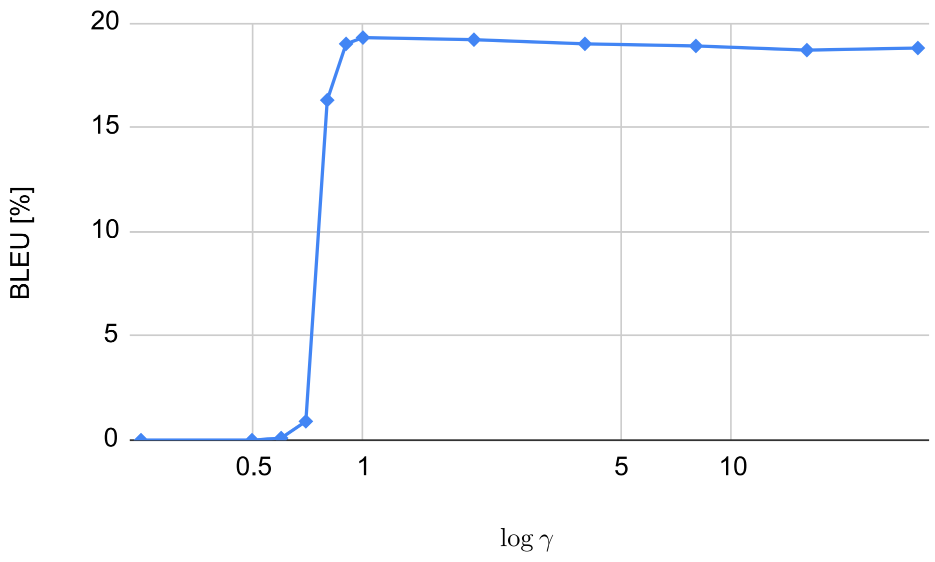


Figure 5.18: Effect of scaling exponent γ for source→pivot posterior probability (WMT 2019 French→German `newstest2011`).

the end task (source→target), the first model must shift away from making actual predictions for pivot language translations of a source sentence. Instead, it is better for the first model to pass sufficient uncertainties to the second model.

Modifying Softmax Probabilities. In Figure 5.18, we apply various values of γ (Equation 5.13) to see how the peakness of a source→pivot softmax distribution influences the performance. Reducing γ to below one causes the model to degenerate, because the softmax outputs are converted to an almost uniform distribution, conveying little information to the pivot→target model. Increasing γ over one also worsens the performance, which is due to the nature of beam search. Some words have suboptimal probabilities at a certain position but allow a higher overall probability product at the end of the sequence in the beams; the softmax weights on their own do not contain such information, so for more peaked distributions, the pivot→target model is likely to process the input preponderantly for the highest-scoring words, which might contain erroneous decisions. All in all, it is found to be of no worth to modify the softmax output from the first model.

5.3.4 Comparison among Presented Methods

In the preceding sections, we have presented sequential transfer and enhanced pivoting for non-English language pairs. Here, we compare the best results for each scheme in Table 5.13 to find out best practice. Note that the given data condition is consistent among all schemes, i.e. the same amount of real bilingual datasets were allowed for training. The pivot-based back-translation was included if it improved the results. We observe that the sequential transfer scheme works best for both tasks; for the performance, tuning for the end task appears to be important.

Table 5.13: Comparison of cross-lingual methods for non-English translation tasks. Every method uses all three types of bilingual data: source-target, source-pivot, and pivot-target.

(a) WMT 2019 German→Czech

	`newstest2012`		`newstest2013`	
Method	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
Baseline	15.7	76.5	18.5	72.0
Multilingual	14.9	76.6	16.5	73.2
Sequential transfer	**18.0**	**72.7**	**21.3**	**68.0**
Pivoting (Cascaded)	18.0	73.6	21.3	68.8

(b) WMT 2019 French→German

	`newstest2012`		`newstest2013`	
Method	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
Baseline	21.1	68.2	22.6	68.1
Multilingual	19.9	69.4	21.7	69.2
Sequential transfer	**21.9**	**67.6**	**23.4**	**67.4**
Pivoting (Cascaded)	21.2	68.4	22.9	68.0

5.3.5 Comparison to Other Work

Most other work investigates non-English tasks in a zero-shot setting where bilingual data is assumed to be nonexistent. There are two reasons for this: First, it avoids the labor of preparing bilingual data in consideration, which becomes important when we want to expand MT to numerous non-English language pairs. Second, it is scientifically more challenging and thus attracts more academic interest. Table 5.14 compares the best outcome of this thesis on the French→German task to recent publications. Both comparative works are based on a multilingual model, while [Arivazhagan & Bapna$^+$ 19] adopt an adversarial loss and [Siddhant & Bapna$^+$ 20] pre-train the model with huge monolingual data. Nevertheless, they all underperform our results. This shows the absolute importance of real bilingual data and sequential transfer for a target language pair in performance. However, the multilingual model with zero-shot translation is still an interesting concept in the light of simplicity in deployment.

Table 5.14: Comparison to other work on NMT for non-English language pairs (French→German).

Data Size (#sents)							`newstest2013`
fr-de	fr-en	en-de	fr	en	de	Method	Bleu [%]
2.5M	35M	9.1M	-	-	-	This thesis	**23.4**
-	39M	4.5M	-	-	-	[Arivazhagan & Bapna$^+$ 19]	20.3
-	39M	4.5M	161M	200M	270M	[Siddhant & Bapna$^+$ 20]	18.5

5.4 Conclusion and Contributions

This chapter covered cross-lingual transfer for NMT models, i.e. the use of pre-trained NMT models of related language pairs (parent) in the final translation task (child). For the tasks where the target language is fixed, we extended plain transfer of a parent model with cross-lingual word embedding, noisy pre-training, and parent-child mixed fine-tuning. These extensions together give consistent improvements, e.g. 19.2% to 22.1% Bleu in IWSLT Slovenian→English tst2014 or 8.9% to 14.0% Bleu in Belarusian→English task of [Qi & Sachan$^+$ 18]. We also proved that synthetic bilingual data is not effective in extremely low-resource scenarios, and proper freezing of parameters with a small vocabulary is necessary for successful cross-lingual transfer.

We also developed transfer techniques for pivot-based scenarios with two parent models, each of which shares the source or target language, respectively. Firstly, we proposed to take the encoder from the source→pivot parent model and the decoder from the pivot→target model. To improve the cohesion from the independently pre-trained encoder and decoder, we proposed additional adapter layers, step-wise pre-training, and cross-lingual training of the encoder. Compared to the direct baseline without pre-training and transfer, these techniques improve e.g. from 13.5% to 18.1% Bleu in WMT German→Czech `newstest2013`. We also verified that zero-shot translations are possible with sequential transfer.

For the pivot-based scenario, we also extended the conventional cascading of the two parents with new interfaces: the weighted sum of embeddings and decoder states. These interfaces enable integrated training of a cascaded system with a source→target bilingual data. We showed that these extensions can further improve the cascading, e.g. from 20.6% to 21.2% Bleu in WMT French→German `newstest2012`. For best practice, however, sequential transfer is still preferred. Also, we showed that synthetic bilingual data is a must-have factor for sequential transfer but not beneficial for the cascaded fine-tuning. In the following, we summarize the credits and publications of this chapter's work.

To-English Tasks. The sequential transfer techniques in Section 5.2.2 were devised and implemented by Yunsu Kim. He also prepared datasets and conducted most of the experiments. Yingbo Gao reviewed the core concepts of the techniques and ran the main experiments for the Belarusian→English and Slovenian→English tasks. The results were published in ACL 2019 [Kim & Gao$^+$ 19] and the implementation was made public[3]. The techniques were also tested in the

[3]`https://github.com/yunsukim86/sockeye-transfer`

WMT 2019 Kazakh→English news translation task dataset and proved to be effective [Rosendahl & Herold$^+$ 19].

Non-English Tasks. Section 5.3.2 is based on the Master's thesis of Petre Petrov [Petrov 19], which was supervised by Yunsu Kim and Shahram Khadivi. Yunsu Kim proposed the topic and designed all presented methods and experiments. Petre Petrov specified the details of the experiments, did all implementations, carried out all experiments, and suggested the use of noisy inputs in the cross-lingual encoder (Section 5.3.2). Shahram Khadivi participated in discussions and analyses with crucial comments. The results of the sequential transfer methods for non-English translation tasks were published in EMNLP 2019 [Kim & Petrov$^+$ 19].

Section 5.3.3 is based on the Bachelor's thesis of Benedikt Hilmes [Hilmes 20], which was supervised by Yunsu Kim. Yunsu Kim developed the presented methods, designed most of the experiments, and directed all discussions and analyses. Benedikt Hilmes implemented the presented methods and conducted all experiments. He also developed the connection variants of the decoder state interface (Section 5.3.3) and devised the analysis of loss curves (Figure 5.17).

6. Unsupervised Learning

Finally, this chapter is devoted to an extreme data condition for MT, where a translation model is trained solely with source and target monolingual data. Such a scenario is called unsupervised in the sense that the monolingual data is unlabeled, while the bilingual data is basically source text labeled with target translations. Unsupervised MT gains independence from human-translated bilingual data, which is rare and costly to obtain, especially for low-resource language pairs, e.g. Kazakh↔English. For zero-resource language pairs, which are very common among millions of possible language pairs, unsupervised learning is another choice alongside cross-lingual zero-shot translation or pivot-based synthetic data generation (Chapter 5).

There are many variants, but the basic principle of unsupervised MT is identical: we regard the target translation of a source sentence as a latent variable, or vice versa, and improve the hypothesis iteratively with a better model during training. Compared to bilingual data, monolingual data is easy to collect, e.g. via web crawling, and its amount is virtually unlimited for many languages and domains. We hope that this richness in empirical data distribution leads to successful training of an MT model in an unsupervised way.

For low-resource scenarios, the main goal of unsupervised MT is to surpass the performance of (rather weak) supervised learning. It is also important to find out under which conditions this is realized. This chapter contains the attempts to achieve this goal which connect the classical decipherment with the modern unsupervised NMT by gradually improving the modeling. We begin with a word-based decipherment framework and extend it to handle a realistic MT problem using scalability tricks and neural network lexicons (Section 6.2). Then we adopt more modern neural network models, such as cross-lingual embedding and a denoising autoencoder, which slightly modify the previous model combination and introduce reordering (Section 6.3). In the end, we investigate methods to integrate all of the neural components into a single sequence-to-sequence model and analyze its end-to-end training procedure (Section 6.4).

6.1 Related Work

Decipherment. Early work on unsupervised sequence learning was mainly for *deterministic decipherment*, a combinatorial problem of matching input-output symbols with 1:1 or homophonic assumption [Knight & Yamada 99, Knight & Nair$^+$ 06, Ravi & Knight 11a, Nuhn & Schamper$^+$ 13]. *Probabilistic decipherment* relaxes this assumption to allow many-to-many mapping, while the vocabulary is usually limited to a few thousand types [Nuhn & Mauser$^+$ 12, Dou & Knight 13, Nuhn & Ney 14, Dou & Vaswani$^+$ 15]. There have been several attempts to improve the scalability of decipherment methods: For decipherment using the expectation-maximization algorithm, [Nuhn & Mauser$^+$ 12] and [Nuhn & Ney 14] accelerate hypothesis expansions but do not explicitly solve the memory issue for a large lexicon table. Count-based Bayesian inference [Dou & Knight 12, Dou & Knight 13, Dou & Vaswani$^+$ 15] loses all context information beyond bigrams for the sake of efficiency; it is therefore particularly effective in contextless deterministic ciphers or in inducing an auxiliary lexicon for supervised MT. [Ravi 13] uses binary hashing to speed up the Bayesian sampling procedure, but this shows poor performance in large-scale experiments. In this thesis, we adjust the EM training for the traditional decipherment framework to be applicable to 100k-vocabulary translation scenarios (Section 6.2.2).

For the lexicon model, it has been designed to be simple for efficiency: categorical table [Nuhn & Mauser$^+$ 12] or Dirichlet distribution [Ravi & Knight 11b, Dou & Knight 12, Dou & Knight 13]. These are all based on discrete counts of word-to-word relations and involve some kind of smoothing. There is also work on enriching the lexicon with hand-crafted features [Ravi 13, Dou & Vaswani$^+$ 15, Naim & Riley$^+$ 18], but the additional information is limited by the manual feature design. Section 6.2.1 proposes neural lexicon architectures as alternatives to these approaches.

A decipherment framework resembles tagging with a hidden Markov model (HMM). HMM taggers are often integrated with sparse priors [Goldwater & Griffiths 07, Johnson 07], which is not readily possible in a large vocabulary setting due to the memory bottleneck. For unsupervised tagging, [Tran & Bisk$^+$ 16] replace the emission model of HMMs with neural networks. However, as they keep the generative modeling direction, the neural emission model does not scale up, only showing results in part-of-speech tagging where the target vocabulary is less than a hundred. To elaborate the model, they include source side context in the LM component, which still hinders LM state recombinations and accordingly slows down the forward-backward algorithm. We integrate the extended context in the discriminative lexicon component instead, which solves the efficiency problem and enriches the model naturally at the same time.

Exploiting contextual words in a lexicon model proved to be effective in statistical machine translation [Hasan & Ganitkevitch$^+$ 08, Mauser & Hasan$^+$ 09]. In the speech recognition literature, [Bourlard & Morgan 94] invented the integration of neural network acoustic models into an HMM, which is similar to discriminative lexicon modeling in our work. This has been extensively studied for both FFNN [Mohamed & Dahl$^+$ 12] and RNN [Graves & Mohamed$^+$ 13, Zeyer & Doetsch$^+$ 17]. The same approach has been successfully applied to handwriting recognition [Graves 12, Espanã-Boquera & Castro-Bleda$^+$ 11, Doetsch & Kozielski$^+$ 14].

Neural Network Components. As for a modern neural component that is usable in unsupervised MT, cross-lingual word embedding (Section 6.3.1) has been studied first for a supervised setting with a bilingual dictionary at hand. [Mikolov & Le$^+$ 13] notice that continuous embedding spaces exhibit similar structures across languages, even for distant language pairs like English-Vietnamese. With separately learned monolingual embeddings, they learn a linear mapping from source embedding space to target embedding space using 5k-pair dictionaries. [Xing & Wang$^+$ 15] show that results can be improved by enforcing an orthogonal constraint on the linear mapping.

To remove the constraint of having a bilingual dictionary, [Artetxe & Labaka$^+$ 17] start the mapping learning from cognates or digits which are common in the two languages, later iteratively aligning the word embedding spaces gradually. [Cao & Zhao$^+$ 16] minimize the distribution dissimilarity, e.g. mean and variance, between source and target embeddings. [Zhang & Liu$^+$ 17] adopt adversarial training between the mapped source embedding and the target embedding, yet underperforming a lot from a supervised setup. [Conneau & Lample$^+$ 18] simplify the adversarial training structure with different loss functions for the generator and discriminator. They also regularize the linear mapping to be orthogonal and introduce cross-domain similarity local scaling (CSLS) to handle hubness.

The denoising autoencoder (Section 6.3.3) was first introduced in computer vision [Vincent & Larochelle$^+$ 08]. [Hill & Cho$^+$ 16] design input noise for natural language text (drop words, permute words), on which they learn sentence embeddings using the autoencoder.

Unsupervised Sequence-to-Sequence NMT. [Artetxe & Labaka$^+$ 18] and [Lample & Denoyer$^+$ 18] exploit the two neural components in training a fully-fledged sequence-to-sequence model in an unsupervised way. They adopt iterative back-translation [He & Xia$^+$ 16, Hoang & Koehn$^+$ 18] in a model whose components are shared between source and target languages. This is extended by [Yang & Chen$^+$ 18] (weight sharing), [Sen & Gupta$^+$ 19] (multilingual setup), [Sun & Wang$^+$ 19] (embedding agreement), and [Conneau & Lample 19, Ren & Wu$^+$ 19] (cross-lingual LM initialization). To further improve the performance at the cost of efficiency, [Lample & Ott$^+$ 18], [Ren & Zhang$^+$ 19] and [Artetxe & Labaka$^+$ 19] combine unsupervised NMT with unsupervised phrase-based MT.

However, these methods are verified mostly in high-resource language pairs, e.g. French↔English, where there is no need to restrict the training data to only monolingual corpora. In low-resource language pairs with little linguistic similarity, [Neubig & Hu 18] and [Guzmán & Chen$^+$ 19] show that unsupervised NMT methods do not function at all.

6.2 Word-based Decipherment

The task of unsupervised MT can be regarded as translating a completely unknown language, where we do not have any knowledge of its relation to other languages. A simple form of such a problem is cracking a substitution cipher, where each word in a given ciphertext (source) corresponds to a word in the plaintext (target) vocabulary (Figure 6.1). The task is to find a mapping between words across the two texts and usually relies on a large plaintext to improve the mapping. The task can be categorized by an assumption on the mapping: each plaintext word is matched with a single ciphertext word (*1-1*) or multiple ciphertext words (*homophonic*). In this thesis, we focus on a more generalized problem: a *probabilistic* substition cipher. Now a plaintext is encoded in such a way that each word is perturbed by a probabilistic distribution $p(f|e)$, possibly switched to any word in the ciphertext vocabulary. Accordingly, we release the constraint of a deterministic mapping and learn a probability model $p(f|e)$ of word-to-word translations, which is called a *lexicon*: the most elementary form of translation modeling.

Note that, here, there is no reordering between source and target words and thus the length does not change from input to output ($I = J$). The most general translation task allows all kinds of reordering, which had been avoided in unsupervised MT due to the huge latent variable space. The reordering-free translation may lack fluency but may still be useful in those scenarios where adequacy is a dominating factor, e.g. translating short messages or text summaries. As for including alignment models in the decipherment framework, we refer the reader to [Nuhn 19].

6.2.1 Model

For the classical decipherment, we model a joint probability of source and target texts similarly to in Section 3.2.1. This enables use of a target LM without heuristics and associates nicely with the maximum likelihood training with unknown labels (Section 6.2.2). For the lexicon modeling, we may choose either a generative or discriminative direction. As for the actual architecture of the lexicon, we investigate both count-based tables and neural network models.

$$\begin{array}{rcccccc}
\text{source sentence } f_1^J = & f_1 & ... & f_j & ... & f_J \\
 & | & & | & & | \\
\text{target sentence } e_1^J = & e_1 & ... & e_j & ... & e_J
\end{array}$$

Figure 6.1: Decipherment task setup with 1:1 monotone alignment between source and target words.

Lexicon Direction

Generative Modeling. The most straightforward way of modeling the joint probability is to decompose it into a target LM and a target-to-source model, just like a noisy channel (Equation 3.9). We apply a further assumption that each source word is *generated* solely from the target word at the same position:

$$p(e_1^J, f_1^J) = p(e_1^J) \cdot p(f_1^J|e_1^J) \tag{6.1}$$

$$= \prod_{j=1}^{J} p(e_j|e_{j-n+1}^{j-1})\, p(f_j|e_j) \tag{6.2}$$

where the LM is also decomposed over the positions with an n-gram approximation. The LM can be trained independently with large monolingual data; it is supposed to compensate for the lexicon and improve the fluency on the target side. This formulation is analougous to the $(n-1)$-th order HMM.

Discriminative Modeling. We also consider the opposite direction of lexicon modeling, i.e. source-to-target: $p(e|f)$. We call this model *discriminative* since it actually discriminates probable output words given an input, rather than assuming generation of input words from a correct output. Lexicon models of this direction are integrated into the joint probability (Equation 6.1) as follows:

$$p(e_1^J, f_1^J) = \prod_{j=1}^{J} p(e_j|e_{j-n+1}^{j-1})\, \frac{p(e_j|f_j)^{\lambda_{\text{LM}}}}{p(e_j)^{\lambda_{\text{prior}}}} \tag{6.3}$$

Since this modeling breaks the correct decomposition of a joint probability, we divide the lexicon model with an output prior distribution $p(e)$ to complement the normalization [Bourlard & Morgan 94]. With the prior, the discriminative lexicon is proportional to the generative lexicon, i.e. $p(e|f)/p(e) \propto p(f|e)$. λ_{prior} and λ_{LM} are scaling parameters which allow us to control the impact of the prior and the LM. They are crucial to stabilize the convergence of the training and optimize the performance. With discrimative modeling, we generally expect better performance with lower asymptotic error compared to the generative model [Schwarz 78, Vapnik 98], as supported by experiments in Section 6.2.4.

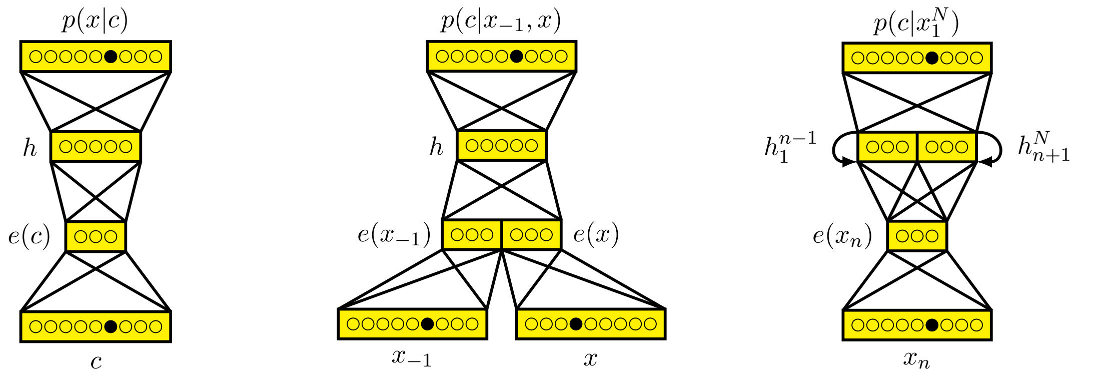
Furthermore, in a discriminative lexicon, we can easily condition on surrounding input words to enlarge the contextual information, e.g. $p(e_j|f_j, f_{j-1})$ [Hasan & Ganitkevitch$^+$ 08, Mauser & Hasan$^+$ 09]. To make the best use of contextual information, we may also use an entire source sentence as the input: $p(e_j|f_1^J)$. Note that the prior stays the same in the denominator for the contextualized lexicon models.

Lexicon Architecture

Count-based Table. In a naive way, a lexicon model can be parametrized by a categorical table over the entire source and target vocabularies:

$$p(f|e) = \theta_{f|e} \tag{6.4}$$

with normalization constraints $\forall_e \sum_f \theta_{f|e} = 1$. Each entry in the table represents a normalized count of a source word f being mapped to a target word e, which is estimated in training. The definition for the discriminative case is analogous. This is simple to implement and fast in query ($O(1)$), but the model size grows exponentially over the vocabulary size ($O(V_f V_e)$).



(a) Generative direction without context.

(b) Discriminative direction with a predecessor context.

(c) Discriminative direction with full context.

Figure 6.2: Neural network lexicon model architectures. The bottommost are input layers and the topmost are output layers. e and h denote embedding and hidden layers, respectively. Circles in each layer indicate the dimensions, where the concerned is filled with black.

Neural Network. A table cannot model more than frequency counts, while a neural network potentially learns complicated information with built-in regularization (Section 3.3). Moreover, a neural lexicon can have a flexible model capacity which can cover large vocabularies with low memory requirement, which is essential in decipherment. Here, we present three neural architectures for a word-based lexicon.

We first suggest a simple feedforward network to model the word-to-word relation. For example, $p(f|e)$ is modeled by a series of layers introduced in Section 3.3.1: a one-hot encoding of e, source embedding, a couple of hidden layers, and an output layer for predicting f with softmax activation (Figure 6.2a). In the context of speech or handwriting recognition, this generative direction is impossible in hybrid HMM [Bourlard & Morgan 94], where the output is a continuous feature set whose inference is not trivial by neural networks. Lexicon modeling with neural networks has no such problem in both generative and discriminative models.

Compared to the table model, the most notable feature of neural network lexicon models is painless integration of source side context. We can easily extend the feedforward models to depend on more input words like [Schwenk 12]. For example, when modeling with a predecessor word, i.e. $p(e|f_{-1}, f)$, we simply feed f_{-1} and f together into the input layer, concatenate their embeddings, and pass the concatenated representation to a hidden layer (Figure 6.2b). Since all source words share the same embedding lookup table, the model complexity is not increased from the input layer to the embedding layer. However, the table model size blows up even with a single contextual word, which may require harsh pruning of the table entries (Section 6.2.2).

The feedforward structure cannot afford a long context, e.g. 10 words, since the output of an embedding layer becomes too large, making the hidden layer computation too expensive. Therefore, we adopt a recurrent neural network (RNN) to store an unlimited context in a fixed size hidden state. The hidden state of the previous (or next) position is fed back to the computation of the current hidden state, recursively encoding a context up to the current position. Figure 6.2c shows a bidirectional RNN lexicon model, which has two recurrent units: a forward unit for past context and a backward unit for future context. The forward and backward hidden states are put together to the next layer. By adjusting the layer sizes or using a recurrent unit, we are able to keep the model size reasonably small for loading in memory.

In the discriminative direction, a neural network lexicon outputs a distribution over all target words, which can be reused in all hypothesis expansions in the same position (with the same f). In contrast, a generative neural lexicon produces a score for one target word at a time (used as input to the network), which can be used in only one hypothesis expansion with that specific target word. This means that for each position of the forward-backward lattice, we need to perform a neural network inference for every single e if we use the generative lexicon.

6.2.2 Training

Training for decipherment is based on the view of translation output (target) as a latent variable. We first consider the maximum likelihood of the input text (source) alone, and then introduce the target text within a joint probability:

$$\begin{aligned} \mathrm{L}(\theta) &= \sum_{f_1^J \in \mathcal{C}_f} \log p(f_1^J) && (6.5)\\ &= \sum_{f_1^J \in \mathcal{C}_f} \log \sum_{e_1^J} p_\theta(f_1^J, e_1^J) && (6.6) \end{aligned}$$

As training data we are given only source monolingual data $\mathcal{C}_f$. We assume that the LM is trained with target side monolingual data beforehand. In this section, we present how to train the lexicon model component, depending on its architecture and modeling direction.

Expectation-Maximization

The expectation-maximization (EM) algorithm [Dempster & Laird+ 77] is a standard method to find a local optimum of maximum likelihood with latent variables. Instead of the original training criterion (Equation 6.6), the EM algorithm defines a weighted sum of the log likelihood as an auxiliary objective:

$$Q(\tilde{\theta}, \theta) = \sum_{f_1^J \in \mathcal{C}_f} \sum_{e_1^J} p_\theta(e_1^J | f_1^J) \cdot \log p_{\tilde{\theta}}(f_1^J, e_1^J) \qquad (6.7)$$

where $\tilde{\theta}$ is a new, unknown parameter set and θ is the old parameter set from the previous iteration. The weight is a posterior probability of the latent target translation computed with the current parameter. The algorithm optimizes this quantity with respect to the new parameter set $\tilde{\theta}$:

$$\begin{aligned} \underset{\tilde{\theta}}{\operatorname{argmax}} \left\{ Q(\tilde{\theta}, \theta) \right\} &= \underset{\tilde{\theta}}{\operatorname{argmax}} \left\{ \sum_{f_1^J \in \mathcal{C}_f} \sum_{e_1^J} p_\theta(e_1^J | f_1^J) \cdot \sum_{j=1}^{J} \left[\log p(e_j | e_{j-n+1}^{j-1}) + \log p_{\tilde{\theta}}(f_j | e_j) \right] \right\} && (6.8)\\ &= \underset{\tilde{\theta}}{\operatorname{argmax}} \left\{ \sum_{f_1^J \in \mathcal{C}_f} \sum_{j=1}^{J} \sum_{e_1^J} p_\theta(e_1^J | f_1^J) \cdot \log p_{\tilde{\theta}}(f_j | e_j) \right\} && (6.9)\\ &= \underset{\tilde{\theta}}{\operatorname{argmax}} \left\{ \sum_{f_1^J \in \mathcal{C}_f} \sum_{j=1}^{J} \sum_{e \in \mathcal{V}_e} p_{\theta,j}(e | f_1^J) \cdot \log p_{\tilde{\theta}}(f_j | e_j) \right\} && (6.10) \end{aligned}$$

for a generative word-to-word lexicon as an example. Note that the LM is pre-trained and fixed, thus dropped from the optimization. The marginal posteriors $p_j(e|f_1^J)$ are defined as:

$$p_j(e | f_1^J) = \sum_{e_1^J :\, e_j = e} p(e_1^J | f_1^J) \qquad (6.11)$$

$$= \frac{\sum_{e_1^J:\, e_j = e} p(f_1^J, e_1^J)}{\sum_{e_1^J} p(f_1^J, e_1^J)} \tag{6.12}$$

This posterior probability is computed by the forward-backward algorithm in the *expectation* step (E-step).

Once the marginal posteriors are at hand, we take a derivative of $Q(\tilde{\theta}, \theta)$ to get an estimate of the lexicon parameters $\tilde{\theta}$. In the case of a table lexicon with the generative direction, we have the following closed-form solution for $\tilde{\theta}$:

$$\tilde{\theta}_{f|e} := \frac{\sum_{f_1^J \in \mathcal{C}_f} \sum_{j:\, f_j = f} p_j(e|f_1^J)}{\sum_{f_1^J \in \mathcal{C}_f} \sum_{f' \in \mathcal{V}_f} \sum_{j':\, f_{j'} = f'} p_{j'}(e|f_1^J)} \tag{6.13}$$

This is called the *maximization* step (M-step) since we maximize the auxiliary objective to find a better optimum for $\tilde{\theta}$. For detailed derivations, we refer the reader to [Schamper 15]. If the lexicon is a discriminative table, the denominator is computed over target words $e' \in \mathcal{V}_e$ instead of source words in the second sum. For discriminative modeling, we assume that the prior $p(e)$ is not trainable. A simple option is to fix it to a pre-trained unigram LM. An alternative is to update it individually after the M-step of every iteration:

$$p(e) = \sum_{f \in \mathcal{V}_f} p(f) \cdot p(e|f) \tag{6.14}$$

$$= \frac{1}{|\mathcal{C}_f|} \sum_{f_1^J \in \mathcal{C}_f} \frac{1}{J} \sum_{j=1}^{J} p_{\tilde{\theta}}(e|f_j) \tag{6.15}$$

In other words, we retrieve the lexicon probabilities over the training data and average all of them. It requires another pass of reading the training data but does not take much time since it does not generate a decoding lattice.

The EM algorithm alternates between the E-step and M-step until a certain stopping criterion is met. Note that the algorithm is sensitive to the initial values for the parameters, since it updates the parameters based on the current values of the model parameters. A common strategy to obtain a good initialization is to randomly sample the parameter values with different random seeds [Berg-Kirkpatrick & Klein 13]. One can find the theoretical considerations of why the EM algorithm works for maximum likelihood estimation in [Bishop 06]. The algorithm has been used also in classical supervised MT where the alignment between source and target sentences is the latent variable [Brown & Della Pietra$^+$ 93].

Sparse Table

In a very large vocabulary setup, the EM training has two fundamental problems for a table lexicon. First, a full lexicon table is too large to keep in memory during training. Second, the search space for hypotheses grows exponentially with the vocabulary size, where both memory and time requirements for the forward-backward step explode. As only a few f's may be eligible translations of a target word e, we propose to keep only those entries with a probability of at least τ:

$$\mathcal{F}(e) = \{f \mid \tilde{\theta}_{f|e} \geq \tau\} \tag{6.16}$$

$$p_\mathrm{s}(f|e) = \begin{cases} \dfrac{\tilde{\theta}_{f|e}}{\sum\limits_{f' \in \mathcal{F}(e)} \tilde{\theta}_{f'|e}} & \text{if } f \in \mathcal{F}(e) \\ 0 & \text{otherwise} \end{cases} \tag{6.17}$$

We call this model a *sparse* lexicon, because only a small percentage of the full lexicon is *active*, i.e. has nonzero probability. The thresholding by τ allows flexibility in the number of active entries over different target words. If e has little translation ambiguity, i.e. the probability mass of $\theta_{f|e}$ is concentrated at only a few f's, $p_\mathrm{s}(f|e)$ occupies less memory than other more ambiguous target words. For each M-step update, it reduces its size on the fly as we learn sparser E-step posteriors.

However, the sparse lexicon might exclude potentially important entries in early training iterations, when the posterior estimation is still not reliable. Once an entry has zero probability, it can never be recovered by the EM algorithm afterwards. A naive workaround is to adjust the threshold during the training, but this does not actually help for the performance in our internal experiments. To give a chance to zero-probability translations throughout the training, we smooth the sparse lexicon with a backoff model $p_\mathrm{b}(f)$:

$$p(f|e) = \lambda_\mathrm{s} \cdot p_\mathrm{s}(f|e) + (1 - \lambda_\mathrm{s}) \cdot p_\mathrm{b}(f) \tag{6.18}$$

where λ_s is the interpolation parameter. As a backoff model, we use uniform distribution, unigram of source words, or the Kneser-Ney lower order model [Kneser & Ney 95, Foster & Kuhn$^+$ 06].

The idea of filtering and smoothing parameters in the EM training is relevant to [Deligne & Bimbot 95] and [Marcu & Wong 02]. They leave out a fixed set of parameters for the whole training process, while we update trainable parameters for every iteration. [Nuhn & Ney 14] also perform an analogous smoothing but without filtering, only to moderate the lattice pruning. Note that our work is distinct from the conventional pruning of translation tables in supervised phrase-based MT, which is applied after the entire training.

Word Class Initialization

Apart from the memory problem, it is unavoidable to apply pruning in the forward-backward algorithm for runtime efficiency. Pruning in early iterations, however, may drop the chances of finding a better optimum in later stages of training. One might suggest pruning only for later iterations, but for large vocabularies, a single non-pruned E-step can blow up the total training time. We instead stabilize the training by a proper initialization of the parameters, so that the training is less detrimentally affected by early pruning. We learn an initial lexicon on automatically clustered word classes [Martin & Liermann$^+$ 98], following these steps:

1. Estimate word-class mappings on both sides
2. Replace each word in the corpus with its class, i.e. $f \mapsto c_f$ and $e \mapsto c_e$
3. Train a class-to-class full lexicon with a target class LM using the EM algorithm
4. Convert 3 to an unnormalized word lexicon by mapping each class back to its member words, i.e. $\hat{\theta}_{f|e} := p(c_f|c_e)$
5. Apply the thresholding on 4 and renormalize (Equation 6.17)

where all f's in an implausible source class are left out together from the lexicon. The resulting distribution $p_\mathrm{s}(f|e)$ is identical for all e's in the same target class.

Word classes group words by syntactic or semantic similarity [Brown & deSouza+ 92], which serve as a reasonable approximation of the original word vocabulary. They are especially suitable for large vocabulary data, because one can arbitrarily choose the number of classes to be very small; learning a class lexicon can thus be much more efficient than learning a word lexicon. These have been widely used in the SMT literature as factors in translation [Koehn & Hoang 07a, Rishøj & Søgaard 11] or smoothing the space of model components [Wuebker & Peitz+ 13, Kim & Guta+ 16]. The general idea of learning a good initialization on a smaller model is inspired by [Och & Ney 03] and [Knight & Nair+ 06].

Generalized Expectation-Maximization

Unlike a table model, neural network lexicons have no closed form solution for the M-step. Thus we search for a suboptimal solution with a numerical algorithm in each iteration. Differentiating Equation 6.10 with respect to $\tilde{\theta}$, we get (for the word-to-word discriminative case):

$$\frac{\partial Q}{\partial \tilde{\theta}} = \sum_{f_1^J \in \mathcal{C}_f} \sum_{j=1}^{J} \sum_{e \in \mathcal{V}_e} p_j(e|f_1^J;\theta) \cdot \frac{\partial}{\partial \tilde{\theta}} \log p_{\tilde{\theta}}(e_j|f_j) \tag{6.19}$$

We backpropagate this gradient to train the network $p_{\tilde{\theta}}$. Note that this is essentially cross-entropy training for a neural network with adaptations to the unsupervised translation. That is, it considers not only a single target label, which is regarded as the ground truth, but also all other target labels, and weights each gradient signal with the marginal posterior.

In practice, we take only a part of the sum—a subset of all (f, e) pairs—as a mini-batch and run stochastic gradient descent (see Section 3.3.2). Each M-step training is limited to a fixed number of epochs, avoiding overfitting to the current posterior statistics. Since an M-step does not fully optimize the auxiliary objective, it is a case of *generalized expectation-maximization* [Dempster & Laird+ 77]. The algorithm is analogous to other architectures of neural lexicons. We can also formulate an online version of the algorithm, i.e. let each EM iteration handle a small part of the training data, but its performance was empirically worse in our internal experiments, so we excluded it in this thesis.

6.2.3 Decoding

Having trained the model, the best hypothesis $\hat{e}_1^J$ can be obtained by two decision rules. The first way is to maximize the joint probability:

$$f_1^J \mapsto \hat{e}_1^J(f_1^J) = \underset{e_1^J}{\operatorname{argmax}} \left\{ p(e_1^J, f_1^J) \right\} \tag{6.20}$$

This can be derived from the Bayes' decision rule with 0-1 string error loss. An efficient method to find $\hat{e}_1^J$ is the Viterbi algorithm. Another way is to maximize the position-wise sum of marginal posteriors:

$$f_1^J \mapsto \hat{e}_1^J(f_1^J) = \underset{e_1^J}{\operatorname{argmax}} \left\{ \sum_{j=1}^{J} p_j(e|f_1^J) \right\} \tag{6.21}$$

This is also based on the Bayes' decision rule, but using 0-1 symbol error loss. The marginal posterior probabilities can be calculated by the forward-backward algorithm. Detailed derivations for these decision rules can be found in [Nuhn 19].

In our experiments we always used the position-level decision rule, because: 1) We empirically found that it usually produces better sequences with higher model scores. 2) During training,

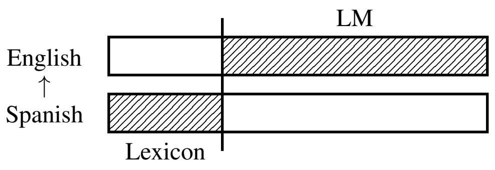


Figure 6.3: Division of a parallel corpus for a decipherment task. Hatched is training data for the corresponding model, while blank is the portion that is not used in training.

the marginal posteriors are calculated in the E-steps. We can reuse the posteriors in decoding the source text to check the intermediate error rates. These advantages are also mentioned in [Johnson 07].

6.2.4 Experiments

We empirically evaluate and analyze the presented decipherment methods in this section. In accordance with the assumption of decipherment, we have chosen simplified translation tasks which force 1:1 monotone alignment between source and target words (Figure 6.1). This shows the performance solely in lexicon training, which is simple yet the most important in delivering semantics to another language. If the reordering problem is included in evaluation, the presented methods may fail equally, regardless of their variations, which hinders us from checking the incremental development of elementary unsupervised MT. Note that the reordering-free translation provides still meaningful semantic clues in a zero-resource condition, though less structured. We may expect more readable outputs when the source and target languages are similar in syntax.

We prepared such a task from the EuTrans Spanish→English corpus. We rearranged the source words of the original parallel corpus to be monotonically aligned to the target words and removed multi-aligned or unaligned words, according to the word alignments learned with GIZA++ [Och & Ney 03]. The corpus was then divided into two parts, using the source text of the first part as an input and the target text of the second part as LM training data (Figure 6.3). In the end, we are given only the monolingual part of each side, which is not sentence-aligned. Note that there is no distinction between training and test sets, where the source dataset is considered as the only ciphertext and used for both the model training and the evaluation. The final corpus statistics are given in Section B.12.

To evaluate a translation output $\hat{e}_1^J$, we use token-level accuracy:

$$\text{Accuracy} = \frac{\sum_{j=1}^{J} [\hat{e}_j = e_j]}{J} \tag{6.22}$$

where e_1^J is the reference output which is the target text of the first division of the corpus. It aggregates all true/false decisions on each word position, comparing the hypothesis with the reference. This can be regarded as the opposite of word error rate (WER) without insertions and deletions. Compared to the metrics of Section 3.6.3, it is more straightforward and natural for a word-by-word translation task.

Table Lexicon: Sparseness and Initialization. First of all, we demonstrate the performance of the classical table lexicon and the effect of the training techniques on it. Table 6.1 reports the decipherment results with full and sparse tables. Having tuned the threshold to be small enough, the sparse lexicon surpasses the performance of the full lexicon, while the number of active entries, for which memory is actually allocated, is greatly reduced. For the backoff, the uniform model shows the best performance, which requires no additional memory. The time complexity is not increased by using the new lexicon.

Table 6.1: Effect of sparse table in word-based decipherment. $\tau = 10^{-3}$ and $\lambda = 0.99$ for the EuTrans task.

Training	Back-off Model	EuTrans Accuracy [%]	EuTrans Table Size [%]
Full	-	70.2	100.0
Sparse	Uniform	72.3	6.3
	Unigram	71.2	6.2
	Kneser-Ney	**72.1**	**6.4**

We also study the mutual effect of τ and λ (Figure 6.4). For a larger τ (circles), where many entries are cut out from the lexicon, the best-performing λ gets smaller ($\lambda = 0.1$). In contrast, when we lower the threshold enough (squares), the performance is more robust to the change of λ, while a higher weight on the trained lexicon ($\lambda = 0.7$) works best. This means that, the higher the threshold is set, the more information we lose and the backoff model plays a bigger role, and vice versa.

Table 6.2 shows that translation quality is consistently enhanced by the word class initialization, which compensates for the performance loss caused by harsh pruning. With a larger number of classes, we have a more precise pre-estimate of the sparse lexicon and thus have more performance gain. Due to the small vocabulary size, we are comfortable with using a higher order class LM, which yields even better accuracy, outperforming the non-pruned results of Table 6.1. The memory and time requirements are only marginally affected by the class lexicon training. Empirically, we find that the word classes do not really distinguish different conjugations of verbs or nouns. Even if we increase the number of classes, they tend to subdivide the vocabulary more based on semantics, keeping morphological variations of a word in the same class. From this fact, we argue that the word class initialization can be generally useful for language pairs with different roots. We also emphasize that word classes are estimated without any model training or language-specific annotations. This is a clear advantage for unknown/historic languages, where the unsupervised translation is indeed in need.

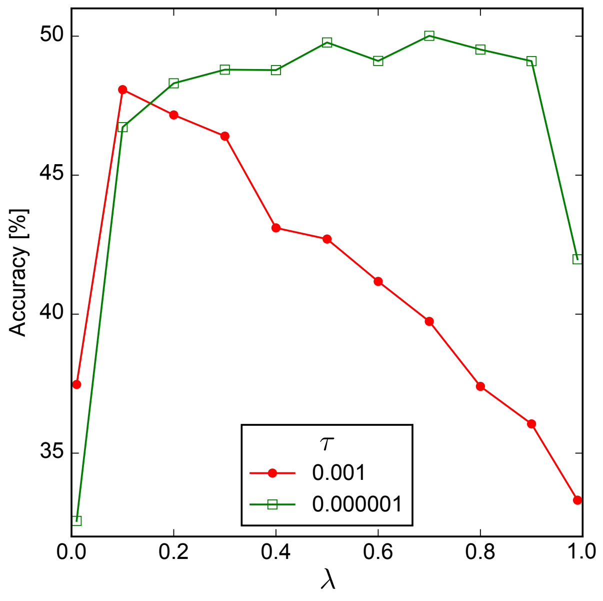


Figure 6.4: Effect of sparse table threshold parameter on translation accuracy.

Table 6.2: Sparse lexicon with word class initialization ($\tau = 0.001$, $\lambda = 0.99$, uniform backoff). Pruning is applied to E-step posteriors with histogram size 10.

Initialization			Accuracy [%]
			EuTrans
Uniform			63.7
	#Classes	Class LM	
Word Classes	25	2-gram	67.4
	50	2-gram	69.1
	100	2-gram	72.1
	50	3-gram	76.0
	50	4-gram	**76.2**

Neural Network Lexicon: Context and Speed. Next, we analyze with neural network based lexicons. In our experiments, the networks have no embedding layer but one hidden layer of 100 dimensions, and the network weights were initialized by random sampling from a standard Gaussian distribution. For each M-step, we performed 2 epochs of Adadelta [Zeiler 12] with a batch size of 256. Our training algorithm converged after around 50th iterations, of which we show the error rates. We found that convergence is quite slow and often sticks in a bad local optimum if we train from scratch. To have an advanced starting point, we pre-train the network with the posteriors extracted from a fully trained table model for 20 epochs. Note that the entire training procedure is still unsupervised. In all cases, we do not apply any kind of posterior pruning.

Table 6.3 summarizes the results of discriminative neural lexicons with additional source context. The discriminative word-to-word lexicon outperforms the table model results of Table 6.1. Adding source context does not help the translation but rather decreases the accuracy. The reason could be that the lexicon does not distinguish the input words, i.e. which word corresponds to the target output at the current position and which words are context words. A possible solution would be to include positional information or have a weighting mechanism for the input words, which are pervasive in modern sequence-to-sequence models. We will investigate unsupervised learning for the sequence-to-sequence NMT in Section 6.4.

Table 6.4 compares the decoding speed of generative and discriminative neural lexicons. The discriminative case is around three times faster on the EUTRANS task. As explained in Section 6.2.1, this is because we only need to do a single forward step per position with the discriminative lexicon, whereas a generative lexicon must be queried as many times as the target vocabulary size for each position. Here, the speed measure is not exactly scaled by the vocabulary size, since there are many other running time factors in the forward-backward algorithm, e.g. hypothesis expansion, LM state recombination, etc. In addition, the discriminative modeling is amenable to maintaining a cache of softmax outputs for more speedup. The cache is indexed by source words and queried for every position. If the cache contains an entry for the current source word, we retrieve the softmax output directly without network computations. Various strategies are

Table 6.3: Comparison among the context spans of discriminative neural network lexicon for word-based decipherment.

Lexicon Formulation	EuTrans
	Accuracy [%]
$p(e\|f)$	**73.3**
$p(e\|f, f_{-1})$	71.6
$p(e\|f, f_{-1}, f_{+1})$	71.0

Table 6.4: Comparison of decoding speed between generative and discriminative neural lexicons.

Modeling	EuTrans
Direction	Speed [words/sec]
Generative	2.2
Discriminative	**6.4**

available for selecting the cache candidates: frequent words, recently queried words, or likely words in the current domain, etc. The cache size can be flexible to the given memory capacity. For the results in Table 6.4, we used a cache for all preceding source words in each sentence. The generative direction can also keep such a cache, which would, however, have only a marginal effect if the target vocabulary is much larger than the cache size. The table model has no difference in decoding speed, where the lexicon query is always $O(1)$ regardless of the modeling direction.

Comparison of Lexicon Models. Table 6.5 summarizes the decipherment results of lexicons with different modeling directions and architectures. Here, the supervised learning scores were obtained by decoding with an optimal lexicon estimated from the input text and its reference. With the help of word class initialization, table models perform better than neural lexicons, which we found more difficult to train stably. Also, discriminative modeling proves to be more effective than its generative counterpart. Note that it is still far from the performance of supervised learning with decipherment methods, even though we restrict the task setting to word-by-word translation.

Table 6.5: Comparison of lexicon modeling schemes for word-based decipherment.

	Modeling	Lexicon	EuTrans
Learning	Direction	Architecture	Accuracy [%]
Supervised	Generative	Table	97.9
Unsupervised	Generative	Table	76.2
		Neural	70.7
	Discriminative	Table	**76.5**
		Neural	73.3

6.3 Neural Cascaded Model Combination

A decipherment model (Section 6.2.1) is factorized clearly, originating from the noisy channel framework. Its training is also mathematically well-defined with rigorous derivations of the EM algorithm (Section 6.2.2). However, the lexicon does not fully exploit the monolingual data of both sides and learns only from the frequency of source inputs and hypothetical target outputs. Moreover, since the reordering problem is ignored, the output may have a barely acceptable fluency. There were attempts to include source-target word alignments in a decipherment model [Ravi & Knight 11b, Nuhn 19], but these were limited to only elementary and local distortions while significantly increasing the complexity of training and decoding. These points are the reasons why word-based decipherment has a limited performance.

In this section, we break away from the rigid formulation of decipherment models and show a novel combination of modern neural components for an improved unsupervised MT. Section 6.3.1 introduces cross-lingual word embedding, which learns rich context-based representations of words from both source and target monolingual data while sharing the mathematical space across the two languages. Section 6.3.2 combines a target LM with the cross-lingual word embedding to improve the translation performance in context. Finally, Section 6.3.3 presents the denoising autoencoder, which is an end-to-end model for implementing reordering directly at the sequence level. These sections are based on recent literature [Conneau & Lample$^+$ 18, Hill & Cho$^+$ 16] with original extensions by this thesis.

6.3.1 Cross-lingual Embedding

Word embeddings define distributed representations of words in a vector space (Section 3.3.1). In a word embedding space, semantically close words have similar representations, building a geometry of word relations. [Mikolov & Le$^+$ 13] observe that such geometries exhibit isomorphic structures across languages, even if they are trained separately on monolingual corpora of each language. When sharing an embedding space among multiple languages, it is possible to query a semantic counterpart of a word in another language by distance calculations.

This opens an alternative to feedforward/recurrent neural lexicons (Section 6.2.1) in word-to-word translation. Instead of training a network directly with source-target word pairs, we train word embeddings with context information using monolingual data and realize the sharing among the vector spaces. This can make use of large monolingual resources equally for each language, encoding rich contextual semantics on both sides. The cross-lingual sharing of embeddings has already been presented in Section 5.2.2 as a bridge in sequential transfer of sequence-to-sequence NMT models, but here we build the embedding explicitly for word translation and introduce original contributions: corpus-based training and cross-lingual phrase embedding.

Model

As in Section 5.2.2, we use a mapping-based cross-lingual word embedding method. We assume that monolingual word embedding is pre-trained for each language in advance and the cross-linguality is modeled with a linear mapping $\mathbf{W}_{f,e}$ from one language to another, e.g. source embedding to target embedding. Then the mapped source embedding $\mathbf{W}_{f,e} \cdot \mathrm{b}_f(f)$ is supposed to be distributed in the same space as the target embedding $\mathrm{b}_e(e)$. The geometric mapping between two embedding spaces might not necessarily be linear, but a non-linear mapping has been shown to be no better than the linear case [Mikolov & Chen$^+$ 13, Conneau & Lample$^+$ 18]. The mapping-based approach is simple and can be done for any new language pair in a short amount of training time, once having pre-trained monolingual embeddings. Other approaches, e.g. joint training without a mapping, are comprehensively explained in [Ruder & Vulić$^+$ 19].

We also use the same linear mapping model on monolingual phrase embeddings to obtain *cross-lingual phrase embedding*. Monolingual phrase embedding can be trained using the same algorithm as for the word embedding, where a phrase is considered as a non-divisible unit. In this thesis, we limit the length of a phrase to bigrams. Before the embedding training, we count the most frequent bigrams in the training corpus, using the counts directly as phrase score:

$$\mathrm{score}(e', e) = \mathrm{count}(e', e) \tag{6.23}$$

Then we maximize the sum of phrase scores in a given sentence using dynamic programming. This method is easy to implement and there is no hyperparameter to tune.

Cross-lingual Search

Given the linear mapping model above, we may use a nearest neighbor search to perform word translation. For example, a target translation e of a source word f is searched as follows:

$$e = \underset{e'}{\mathrm{argmax}} \left\{ \mathrm{sim}(\mathbf{W}_{f\to e} \cdot \mathrm{b}_f(f),\, \mathrm{b}_e(e')) \right\} \tag{6.24}$$

where sim defines a similarity measure between two embedding vectors. This is based on the assumption that the mapped source embedding lies in the same space as the target embedding.

For the similarity measure, basic choices would be cosine similarity:

$$\mathrm{sim}(\mathbf{x},\mathbf{y}) = \cos(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\cdot\|\mathbf{y}\|} \tag{6.25}$$

or (negated) Euclidean distance:

$$\mathrm{sim}(\mathbf{x},\mathbf{y}) = -\|\mathbf{x}-\mathbf{y}\|_2 \tag{6.26}$$

These measures suffer severely from the so-called *hubness* problem, i.e. points tend to be nearest neighbors of many points in a high-dimensional space, which harms the search accuracy. This problem is addressed by applying a corrective metric, e.g. a cross-domain similarity scaling (CSLS) [Conneau & Lample$^+$ 18]:

$$\mathrm{sim}(\mathbf{x},\mathbf{y}) = 2\cos(\mathbf{x},\mathbf{y}) - \frac{1}{\|\mathcal{N}_{\mathbf{y}}(\mathbf{x})\|}\sum_{\mathbf{y}'\in\mathcal{N}_{\mathbf{y}}(\mathbf{x})}\cos(\mathbf{x},\mathbf{y}') - \frac{1}{\|\mathcal{N}_{\mathbf{x}}(\mathbf{y})\|}\sum_{\mathbf{x}'\in\mathcal{N}_{\mathbf{x}}(\mathbf{y})}\cos(\mathbf{x}',\mathbf{y}) \tag{6.27}$$

where $\mathcal{N}_{\mathbf{y}}(\mathbf{x})$ is a set of nearest neighbors of $\mathbf{x}$ among the candidates of $\mathbf{y}$. The average cosine similarity in this set represents the denseness of $\mathbf{x}$, i.e. how much $\mathbf{x}$ can be considered as a (problematic) *hub*. Accordingly, CSLS penalizes the dense pairs and encourages the similarity between isolated pairs, mitigating the hubness problem.

Another practical factor in the cross-lingual search is limiting the search candidates. If we consider only the top frequent words, e.g. 50k, in the target vocabulary as the translation candidates, it reduces the nearest neighbor search time effectively while maintaining a reasonable overall search performance.

Training

Iterative Procrustes Analysis. Similarly to Equation 5.1, training of the cross-lingual mapping $\mathbf{W}_{f\rightarrow e}$ can be basically done by minimizing the mean squared error between the transformed source embeddings and the target embeddings:

$$\hat{\mathbf{W}}_{f\rightarrow e} = \operatorname*{argmin}_{\mathbf{W}}\left\{\sum_{(f,e)\in\mathcal{D}_{\mathrm{f,e}}}\|\mathbf{W}\cdot\mathrm{b}_f(f)-\mathrm{b}_e(e)\|_2\right\} \tag{6.28}$$

where $\mathbf{D}_{f,e}$ is a bilingual dictionary of source-target word pairs which are the translations of each other. [Xing & Wang$^+$ 15] show that the mapping is improved by constraining it to be an orthogonal matrix. This constraint makes the problem *Procrustes analysis*, whose solution can be analytically obtained by singular value decomposition (SVD):

$$\hat{\mathbf{W}}_{f\rightarrow e} = \operatorname*{argmin}_{\mathbf{W}\in\mathcal{O}(D_\mathrm{e})}\left\{\|\mathbf{W}\mathbf{F}-\mathbf{E}\|_F^2\right\} = \mathbf{U}\mathbf{V}^\top \tag{6.29}$$

$$\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \mathrm{SVD}(\mathbf{E}\mathbf{F}^\top) \tag{6.30}$$

where $\mathbf{F}$ and $\mathbf{E}$ are matrices whose columns are embedding vectors of source or target words, respectively, from the pairs in the training dictionary $\mathcal{D}_{f,e}$. $\mathcal{O}(D_\mathrm{e})$ denotes the space of $D_\mathrm{e}\times D_\mathrm{e}$ orthogonal matrices. After we have learned a mapping by solving Equation 6.29, we may further refine the mapping by a self-learning framework [Artetxe & Labaka$^+$ 17]. The idea is to generate a new dictionary based on the learned mapping, which is used to obtain a new mapping again with the same Procrustes solver. We repeat such a process until a convergence criterion is met.

This iterative refinement is especially helpful when the initial dictionary is of very poor quality, e.g. cognates [Artetxe & Labaka$^+$ 17].

To improve the quality of the model-generated dictionary, we consider only mutual nearest neighbors and limit the number of entries in the dictionary to e.g. 5k. We find that mutual neighbors, i.e. the intersection of dictionaries generated by source→target and target→source translations, work better than their union; that is to say, the quality of a dictionary plays a more important role than its quantity.

Adversarial Training. Up to this point, we have been considering a supervised setting with a bilingual dictionary given. However, in unsupervised MT, we assume that there is no bilingual resource at all, where cross-lingual embedding must be also trained without an external dictionary. If we start the training from a random mapping, the word pairs generated from it would be meaningless in terms of translation. [Peirsman & Padó 10], [Hauer & Nicolai$^+$ 17] and [Smith & Turban$^+$ 17] suggest collecting source and target words that are identical in orthography, e.g. digits, which is not strictly unsupervised and relies on the properties of the training corpora.

Instead, [Conneau & Lample$^+$ 18] use language-adversarial training to obtain an initial mapping before the iterative refinement steps. The goal of the adversarial training is to make the model produce embedding vectors whose space is not distinguishable between source and target languages. For this purpose, it introduces a *discriminator* network $p_{\theta_\mathrm{d}}(\zeta|\mathbf{x})$, where $\mathbf{x}$ is an embedding vector and ζ indicates its language, e.g. $\zeta = 1$ for source language and $\zeta = 0$ for target language. The first loss function of the adversarial training encourages the discriminator to detect the language of a given embedding vector correctly:

$$\mathrm{L}_\mathrm{d}(\theta_\mathrm{d}) = -\frac{1}{|\mathcal{V}_f^\mathrm{t}|}\sum_{f\in\mathcal{V}_f^\mathrm{t}} \log p_{\theta_\mathrm{d}}(1|\mathbf{W}\cdot \mathrm{b}_f(f)) - \frac{1}{|\mathcal{V}_e^\mathrm{t}|}\sum_{e\in\mathcal{V}_e^\mathrm{t}} \log p_{\theta_\mathrm{d}}(0|\mathrm{b}_e(e)) \tag{6.31}$$

where $\mathcal{V}_f^\mathrm{t}$ or $\mathcal{V}_e^\mathrm{t}$ is a set of the most frequent words in the source or target language, respectively. Conversely, the second loss function optimizes the model to generate embeddings whose language cannot be inferred easily, eventually promoting a shared embedding space between the two languages:

$$\mathrm{L}_\mathrm{g}(\mathbf{W}) = -\frac{1}{|\mathcal{V}_f^\mathrm{t}|}\sum_{f\in\mathcal{V}_f^\mathrm{t}} \log p_{\theta_\mathrm{d}}(0|\mathbf{W}\cdot \mathrm{b}_f(f)) - \frac{1}{|\mathcal{V}_e^\mathrm{t}|}\sum_{e\in\mathcal{V}_e^\mathrm{t}} \log p_{\theta_\mathrm{d}}(1|\mathrm{b}_e(e)) \tag{6.32}$$

Note that the first loss updates only the discriminator parameters and the second loss only the cross-lingual linear mapping. The two losses are optimized alternately like a two-player game [Goodfellow & Pouget-Abadie$^+$ 14]. In this work, the discriminator is a simple feedforward network with an embedding vector as its input and a one-dimension logit as its output.

During the adversarial training, we perform a proxy manipulation step after every update of the discriminator to make the mapping matrix close to orthogonal [Cisse & Bojanowski$^+$ 17]:

$$\bar{\mathbf{W}} = (1+\beta)\mathbf{W} - \beta(\mathbf{W}\mathbf{W}^\top)\mathbf{W} \tag{6.33}$$

where β is a nonnegative hyperparameter which is set close to 0.

Corpus-based Training. The two presented training schemes are executed by iterating over words in the vocabularies. They use only a single translation (nearest neighbor) per word in learning the cross-lingual mapping, without considering contextual information or polysemy. Also, each word pair contributes equally in training, ignoring its importance or frequency.

To address these issues, we propose *corpus-based* training, in opposition to the above-mentioned vocabulary-based training methods. The idea is to iterate over words in a given corpus and translate them to obtain word pairs for training the cross-lingual mapping. We integrate an LM on the hypothesis side to guide the translation in context, similarly to Section 6.3.2. This is inspired by decipherment models (Section 6.2), where an LM improves the quality of word-by-word translations. This enables bootstrapping out of a random initial mapping, removing the necessity of adversarial training or heuristics for an initial dictionary.

Corpus-based training has no fixed number of word pairs like vocabulary-based methods. By translating a natural corpus, we generate more examples for more frequent words, effectively weighting them more. It goes over all positions in the corpus and uses their translations in updating the mapping on the fly in a mini-batch fashion. Thus it does not have a closed-form solution such as Procrustes analysis but rather resorts to gradient-based stochastic updates.

6.3.2 Context-aware Beam Search

The word translation by nearest neighbor search does not consider the context around the current word. In many cases, the correct translation is not the nearest target word but other close words with morphological variations or synonyms, depending on the context. For example, the German word “verstehen” means “to understand” in most cases. However, Table 6.6 shows the top-5 nearest neighbors in the English vocabulary in a cross-lingual embedding space. The list has also a nominalized form (“understanding”) and a synonym (“comprehend”) of the top candidate “understand”. The correct translation of “verstehen” may be among these, depending on the context. We also observe clearly wrong translations (“verbalize”, “explain”), where context might help to exclude them by giving a lower score. The reasons are two-fold: 1) Word embedding is trained to place semantically related words nearby, even though they have opposite meanings. 2) The hubness problem of a high-dimensional embedding space hinders the correct search where lots of different words happen to be close to each other (Section 6.3.1).

In this thesis, we integrate context information into the word-by-word translation by combining an LM with cross-lingual word embedding. We define the score of a target word e at the current position j as follows:

$$q(e|e_1^{j-1}, f_j) = \log q(f_j, e) + \lambda_{\text{LM}} \log p(e|e_1^{j-1}) \tag{6.34}$$

Here, $q(f, e)$ is a lexical score defined as:

$$q(f, e) = \frac{\cos(\mathbf{W}_{f,e} \cdot \mathrm{b}_f(f),\, \mathrm{b}_e(e)) + 1}{2} \tag{6.35}$$

This is the cosine similarity of the embeddings, transformed to the range $[0, 1]$ to make it similar in scale to the LM probability. In our experiments, we found that this simple linear scaling is better than sigmoid or softmax functions in the final translation performance. The combined score is basically a log-linear model (Section 3.2.2) with an LM and the embedding similarity as its features. We perform beam search over the positions to allow only reasonable translation hypotheses.

Table 6.6: Nearest neighbors of German word “verstehen”.

Word	Cosine Similarity
understand	0.664
verbalize	0.511
explain	0.508
understanding	0.507
comprehend	0.506

Handling Unknown Words. When using cross-lingual word embedding for translation, the vocabulary is usually constrained, e.g. to 50k, in search, because it is infeasible to run a nearest neighbor algorithm for the entire vocabulary, and including rare word embeddings may worsen the hubness problem. As a result, we observe many unknown source words in translation. One can merely generate a pre-defined unknown token, e.g. `<unk>`, but we compare more active solutions. A reasonable approach is to find translation candidates from the LM. For each position, we have already decoded the sentence up to the previous position (e_1^{j-1}), so we could look up the LM table of $p(e|e_1^{j-1})$ and choose the most probable target word e's. In this case, we use only the LM score in such positions for beam search. Another way is to copy the unknown source word to the target side without translating it. This has proved to be effective for named entities that are the same across languages, e.g. person or company names. Using subword units is a more linguistically motivated method. By dividing every word into smaller units, one can effectively reduce the vocabulary size. For an unknown word, we translate subwords of the word and expect the combination of them to be a meaningful translation.

6.3.3 Denoising Autoencoder

Even when we have correctly translated words for each position using cross-lingual embedding and an LM, the output is still far from an acceptable translation since we stick to the 1-1 monotonic alignment. We adopt a sequence denoising autoencoder [Hill & Cho$^+$ 16] to improve the translation output. The idea is to train a sequence-to-sequence neural network model that takes a noisy sentence as input and produces a (denoised) clean sentence as output, both of which are of the same (target) language. The model was originally proposed to learn sentence embeddings, but here we use it directly to actually remove noise in a sentence.

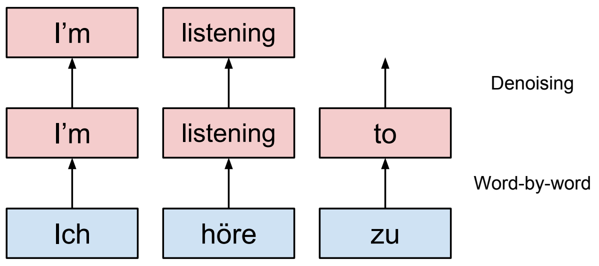
Training label sequences for the denoising network would be target monolingual sentences, but we do not have their noisy versions at hand. Given a clean target sentence, the noisy input should ideally be a word-by-word translation of the corresponding source sentence. However, such bilingual sentence alignment is not available in our unsupervised setup. Instead, we inject artificial noise into a clean sentence to simulate the noise of word-by-word translation. We design different noise types after the following aspects of word-by-word translation.

Insertion. Word-by-word translation always outputs a target word for every position. However, there are a plenty of cases where multiple source words should be translated to a single target word, or where some source words are better not translated to any word to make a fluent output. For example, the German sentence “Ich höre zu.” would be translated to “I'm listening to.” by a word-by-word translator, but “*I'm listening.*” is more natural in English (Figure 6.5a). One might think of having an empty token in the target vocabulary to handle this problem, generating this token for suitable positions. To implement this, however, we need a target corpus including empty tokens for learning to predict such positions. Whether to put an empty token or not can only be learned from word alignments of bilingual sentence pairs, which is again not available in unsupervised MT.

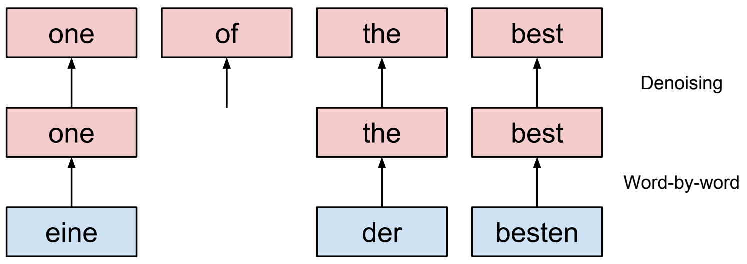
We pretend to have extra target words which might be translations of redundant source words by inserting random target words into a clean sentence:

1. For each position j, sample a probability $p_j \sim \text{Uniform}(0,1)$.
2. If $p_j < p_{\text{i}}$, sample a word e from the most frequent V_{i} target words and insert it before position i.

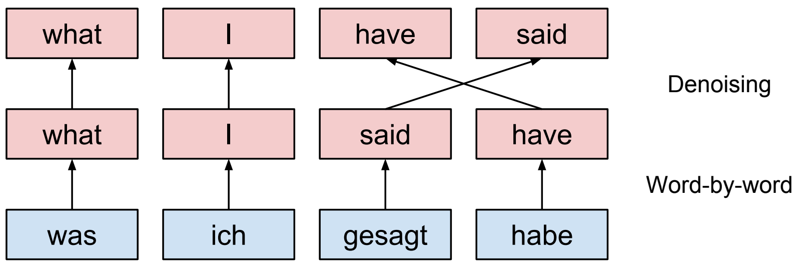
We limit the inserted words by V_{i} because target insertion occurs mostly with common words, e.g. prepositions or articles, as in the example above. We insert words only before—not after—a



(a) Insertion



(b) Deletion



(c) Permutation

Figure 6.5: Examples of denoising.

position, since an extra word after the ending word (usually punctuation) is not probable. The insertion noise is of a design original to this thesis work, which has not been used before.

Deletion. Similarly, word-by-word translation cannot handle the contrary case: when a source word should be translated into more than one target words, or a target word should be generated from no source words for fluency. For example, the German word "im" must be "in the" in English, but word translation generates only one of the two English words. Another example is shown in Figure 6.5b. To simulate such situations, we drop some words randomly from a clean target sentence [Hill & Cho$^+$ 16]:

1. For each position j, sample a probability $p_j \sim \text{Uniform}(0, 1)$.
2. If $p_j < p_\text{d}$, drop the word in the position j.

Permutation. Also, translations generated word-by-word are not in the order of the target language. In our beam search, the LM only assists in choosing the right word in context but does not modify the word order. A common reordering problem of German→English is illustrated in Figure 6.5c. From a clean target sentence, we corrupt its word order by random permutations. We limit the maximum distance between an original position and its new position like [Lample & Denoyer$^+$ 18]:

1. For each position j, sample an integer δ_j from $[0, d_\mathrm{p}]$.
2. Add δ_j to index j and sort the incremented indices $j + \delta_j$ in an increasing order.
3. Rearrange the words to be in the new positions to which their original indices have moved by Step 2.

This is a generalized version of swapping two neighboring words [Hill & Cho$^+$ 16]. Reordering is highly dependent on each language, but we found that this noise is generally close to word-by-word translation outputs.

Insertion, deletion, and permutation noises are applied to each mini-batch with different random seeds, allowing the model to see various noisy versions of the same clean sentence over the epochs. A denoising autoencoder is distinct from reordering features in phrase-based MT, whose weight or the feature itself must be trained on a parallel dataset. Our denoising model addresses various alignment/reordering aspects at the same time while remaining unsupervised. The denoiser is applied directly to the word-by-word translation of Section 6.3.2. Each component in this two-step decoding process (cross-lingual word embedding, LM, denoising autoencoder) is trained individually without an iterative procedure as in decipherment methods.

6.3.4 Experiments

We evaluated the proposed neural components on unsupervised translations for WMT 2018 German→English and WMT 2014 French→English tasks. For German/English, we trained word embeddings with 100M sentences sampled from News Crawl 2014-2017 monolingual corpora. For French, we used News Crawl 2007-2014 (around 42M sentences). The data was lowercased, while German compound words were split beforehand and numbers were replaced with category labels. After translation, we applied truecasing and the numbers were recovered by looking at the source sentence.

FASTTEXT [Bojanowski & Grave$^+$ 17] was used to learn monolingual embeddings only for the words with minimum count 10. MUSE [Conneau & Lample$^+$ 18] was used for cross-lingual mappings with the top 100k frequent words for the adversarial training and 10 Procrustes refinement iterations. Other parameters follow the values in [Conneau & Lample$^+$ 18]. With the same data, we trained 5-gram count-based LMs using KenLM [Heafield 11] with its default setting. Denoising autoencoders were trained using Sockeye [Hieber & Domhan$^+$ 17] on News Crawl 2016 for German/English and News Crawl 2014 for French. We considered only the top 50k frequent words for each language and mapped other words to `<unk>`. The unknowns in the denoised output were replaced by missing words from the noisy input by a simple line search. As a validation set for the denoiser training, we used `newstest2015` (German→English) or `newstest2013` (French→English), where the input/output sides both have the same clean target sentences, encouraging a denoiser to keep at least the clean part of word-by-word translations. Here, the noisy input showed a slight degradation of performance; the model seemed to overfit to specific noises in the small validation set. The total training time of our method is 1-2 days with a single GPU.

In decoding, we used $\lambda_\mathrm{LM} = 0.1$ with beam size 10. The embeddings were mean-centered to zero and normalized to unit length for a consistent search space [Artetxe & Labaka$^+$ 16]. We only translated the top frequent 50k source words and merely copied other words to the target side. For each position, only the nearest 100 target words were considered.

Combination of Neural Components. Table 6.7 shows the performance of combining the proposed neural components. An LM or a denoising autoencoder improves word-by-word baselines consistently in all four test sets, giving at least +3% BLEU. When our denoising model is applied on top of the LM-assisted translation, we have an additional gain of around +3% BLEU. Note that our methods do not involve any decoding steps to generate pseudo-parallel training data. The

Table 6.7: Translation performance of cascaded combinations of unsupervised neural components.

(a) WMT 2018 German→English

Data Size (#sents)			Method	`newstest2016`	
				Bleu	Ter
de-en	de	en		[%]	[%]
5.9M	-	-	Transformer (supervised)	37.2	48.3
-	100M	100M	Word-by-word	11.1	68.0
			+ LM	14.5	65.4
			+ Denoising autoencoder	**17.2**	**64.3**

(b) WMT 2014 French→English

Data Size (#sents)			Method	`newstest2014`	
				Bleu	Ter
fr-en	fr	en		[%]	[%]
35M	-	-	Transformer (supervised)	34.3	52.7
-	42M	100M	Word-by-word	10.6	74.2
			+ LM	13.6	71.3
			+ Denoising autoencoder	**16.5**	**69.4**

performance is still far behind the supervised results, although we used much larger monolingual datasets and the source and target languages are linguistically similar to each other.

Learning Algorithms for Cross-lingual Mapping. To optimize the performance of cross-lingual word embedding, our main component of a neural cascaded system for unsupervised translation, we compare its mapping-based learning algorithms in Table 6.8. The performance of each algorithm is evaluated by translating source words in a reference dictionary and computing their accuracy against their target translations. This intrinsic evaluation rules out the correction effect of other components (LM, denoiser) and helps to focus only on the quality of the cross-lingual mapping. We used a 1500-pair German-English dictionary released by [Conneau & Lample+ 18] for this testing. For the corpus-based algorithm, we ran stochastic gradient descent for 50 iterations with a batch size of 1k sentences a fixed learning rate of 0.0001. As the training corpus, we reused the source monolingual data that was used for learning the source monolingual embedding. We found that using massive data has no significant benefit here; we sampled 50k sentences from the monolingual data for the corpus-based training. Note that we do not need to prepare a training corpus for vocabulary-based algorithms.

Our corpus-based approach surpasses the minimum squared error training and adversarial training based on source/target vocabularies. When an LM is integrated into the corpus translation

Table 6.8: Comparison of mapping learning algorithms for cross-lingual word embedding (German→English).

Type	Method	Accuracy [%]
Vocabulary-based	Minimum squared error	0.0
	Adversarial	53.5
	+ Minimum squared error	**64.9**
Corpus-based	Minimum squared error	54.4
	+ LM in translation	60.4

Table 6.9: Comparison of word and phrase embeddings for cascaded combinations of unsupervised neural components (WMT 2018 German→English).

		`newstest2016`	
		Bleu	Ter
Method	Embedding Unit	[%]	[%]
Word-by-word	Word	11.1	68.0
	Phrase	12.0	67.5
+ LM	Word	14.5	65.4
	Phrase	15.7	66.0
+ Denoising autoencoder	Word	**17.2**	**64.3**
	Phrase	16.9	65.4

process, the performance is significantly boosted further. However, it is worse than vocabulary-based if the adversarial training is followed by the minimum squared error training. We hypothesize that the vocabulary-based adversarial training provides novel calibration of the mapping, which cannot be achieved solely from self-generated word pairs. The frequency-based weighting of training examples in the corpus-based method seems to have no meaningful impact on the performance.

Cross-lingual Phrase Embedding. Table 6.9 showcases the effect of the phrase-level lexicon model. We marked phrases in the training corpus among the top 50k frequent bigrams as described in Section 6.3.1 and train embeddings in the usual way. We also tried threshold-based phrase selection with a discounting coefficient [Mikolov & Sutskever$^+$ 13], but this turned out to be consistently worse than our proposed method. The LM and denoising autoencoder, which eventually deal with both word and phrase entities, were also trained on the same bigram-converted corpus. Phrase-level embedding clearly has an advantage over the usual word-level model in the word-by-word translation thanks to its natural modeling of local context. The superiority holds also for the LM-assisted case, showing +1.2% Bleu improvement against the word-level translation. Nonetheless, cascaded with a denoising autoencoder, phrase-level translation cannot outperform the word-level results. Note that phrase-level translation produces lengthy hypotheses compared to those of the word-level, which explains the higher Ter score in the LM-assisted case despite the improvement in Bleu. We conclude that the word-level denoiser handles the errors in local contexts sufficiently without an explicit phrase modeling, while the phrase-level denoiser occasionally introduces inter-bigram reorderings which are more error-prone.

Artificial Noise Hyperparameters. To examine the effect of each noise type in the denoising autoencoder, we tuned each parameter of the noise and combined them incrementally (Table 6.10). Firstly, for permutations, a significant improvement is achieved from $d_\text{p} = 3$, since a local reordering usually involves a sequence of 3 to 4 words. With $d_\text{p} > 5$, it shuffles too many consecutive words together, yielding no further improvement. This noise cannot handle long-range reordering, which is usually a swap of words that are far from each other, keeping the words in the middle as they are. Secondly, we applied the deletion noise with different values of p_d. 0.1 gives +0.8% Bleu, but we immediately see a degradation with a larger value; it is hard to observe one-to-many translations more than once in each sentence pair. Finally, we optimized V_i for the insertion noise, fixing $p_\text{i} = 0.1$. Increasing V_i is generally not beneficial, since it provides too much variation in the inserted word; it might not be related to its neighboring words. Overall, we observe the best result (+1.5% Bleu) with $V_\text{i} = 50$.

Vocabulary for Cross-lingual Embedding. We also examined how the translation per-

Table 6.10: Translation results with different values of denoising parameters (German→English).

			newstest2016	
d_p	p_d	V_i	Bleu [%]	Ter [%]
2	-	-	14.7	65.3
3			14.9	65.2
5			14.9	65.6
3	0.1	-	15.7	67.6
	0.3		15.1	72.3
3	0.1	10	16.8	65.1
		50	**17.2**	**64.3**
		500	16.8	65.2
		5000	16.5	65.9

Table 6.11: Effect of vocabulary size for cross-lingual word embedding on the translation performance of neural cascaded combination.

Vocabulary		Bleu [%]
	Merges	
BPE	20k	10.4
	50k	12.5
	100k	13.0
	V_t	
Word	20k	14.4
	50k	14.4
	100k	**14.5**
	200k	14.4

formance varies with different vocabularies of cross-lingual word embedding in Table 6.11. The first three rows show that BPE embeddings perform worse than word embeddings, especially with smaller vocabulary size. For short BPE tokens, the context they meet during the embedding training is much more diverse than a complete word, and a direct translation of such a token to a BPE token of another language would be very ambiguous. For word level embeddings, we compared different vocabulary sizes used for training the cross-lingual mapping. Surprisingly, cross-lingual word embedding learned only on the top 20k words is comparable to that of 200k words in translation quality. We also increased the search vocabulary to more than 200k but the performance only degrades. This means that word-by-word translation with cross-lingual embedding depends highly on the frequent word mappings, and learning the mapping between rare words does not have a positive effect.

6.4 Sequence-to-Sequence Model

The cascaded combination in Section 6.3 is fast in training but its two-step decoding is slow. More importantly, the strict two-step process is hard to extend with additional model components. The absence of an iterative training is a plus for quick building of a system, but the performance is limited.

In this section, we present methods to train a sequence-to-sequence model directly in an unsupervised way, where there are no constraints in the alignments or lengths as in Section 6.2. Various concepts like lexical translation, context modeling, and reordering are not handled individually

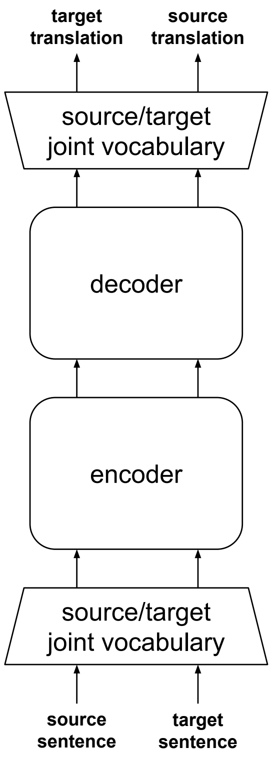


Figure 6.6: Bidirectional modeling of sequence-to-sequence NMT.

as in Section 6.3 but encompassed all together in a single model (Section 6.4.1). The end-to-end model is shown to be better than a combination of individual features in supervised learning, and is also possible to tune without much specific linguistic knowledge. This end-to-end property gives more freedom to the model but makes the training more difficult, so we turn back to an iterative training in a similar spirit to that of the EM algorithm (Section 6.4.2).

The goal of this section is to review the core concepts for unsupervised sequence-to-sequence NMT introduced by [Artetxe & Labaka$^+$ 18], [Lample & Ott$^+$ 18], and [Conneau & Lample 19], and verify its performance in various tasks and data settings (Section 6.4.3). Our extensive empirical results reveal the influence of each component of the method and specify where the unsupervised MT currently is compared to the supervised or semi-supervised learning.

6.4.1 Model

Unsupervised learning is an extreme scenario of MT, where bilingual information is very weak. To supplement the weak and noisy training signal, knowledge transfer and regularization are crucial, which can be achieved by sharing the model parameters between source→target and target→source directions (Figure 6.6). It is also helpful to share a joint subword vocabulary across the two languages [Sennrich & Haddow$^+$ 16c]. The bidirection modeling is a specific case of multilingual NMT (Section 5.3.1), where only two languages are involved with switching translation directions. It is based on the fact that a translation problem is dual in nature; source→target and target→source tasks are conceptually related to each other. Previous works on unsupervised NMT vary in the degree of sharing: the whole encoder [Artetxe & Labaka$^+$ 18, Sen & Gupta$^+$ 19], the middle layers [Yang & Chen$^+$ 18, Sun & Wang$^+$ 19], or the whole model [Lample & Denoyer$^+$ 18, Lample & Ott$^+$ 18, Ren & Wu$^+$ 19, Conneau & Lample 19]. In this thesis, we share the whole model by default. As in the case of multilingual sharing (Section 5.3.1), we inform the decoder of the output language using a language identifier token.

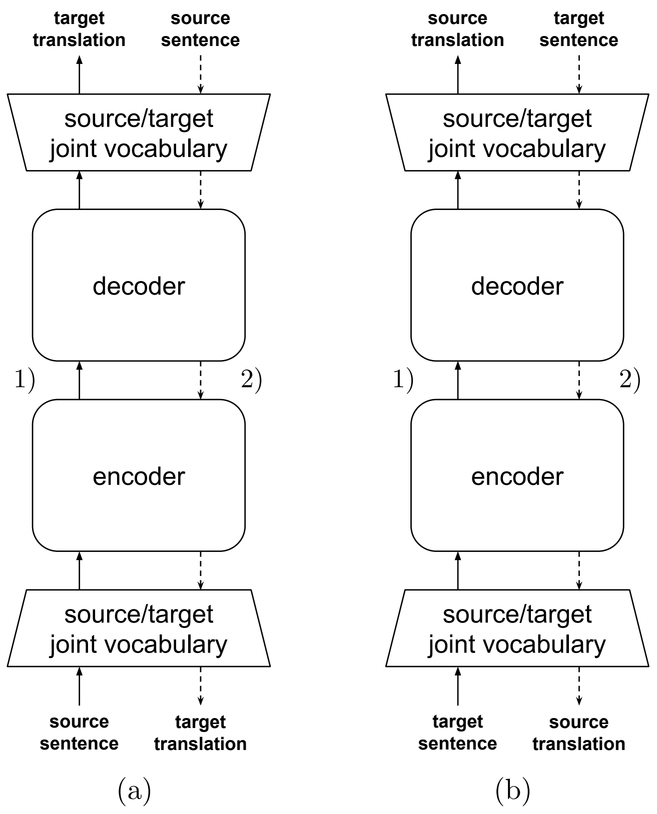


Figure 6.7: Iterative back-translation for training a bidirectional sequence-to-sequence model. The model first translates monolingual sentences (solid arrows), and then gets trained with the translation as the input and the original as the output (dashed arrows). This procedure alternates between (a) source→target and (b) target→source translations.

Note that the network sharing is less effective among linguistically distinct languages in NMT [Kocmi & Bojar 18, Kim & Gao+ 19]. It still works as a regularizer, but transferring knowledge is harder if the morphology or word order is quite different. We show how well unsupervised NMT performs on such language pairs in Section 6.4.3.

6.4.2 Training

Iterative Back-Translation. Unsupervised learning for MT assumes no bilingual data for training. A traditional remedy for the data scarcity is to generate synthetic bilingual data from monolingual text (Section 4.3). To train a bidirectional model (Section 6.4.1), we need bilingual data of both translation directions. Therefore, most unsupervised NMT methods back-translate in both directions, i.e. source and target monolingual data to target and source language, respectively.

In unsupervised learning, the synthetic data must be created not only once at the beginning but also repeatedly throughout the training. In the early stages of training, the model might be too weak to generate good translations. Hence, most methods update the training data as the model gets better during training. The improved model for the source→target direction back-translates the source monolingual data, which improves the model for the target→source direction, and vice versa. This bidirectional, synchronous cycle is called dual learning [He & Xia+ 16] or iterative back-translation [Hoang & Koehn+ 18]. Figure 6.7 shows the case when it is applied to a fully shared bidirectional model. One can tune the amount of back-translations per iteration: a mini-batch [Artetxe & Labaka+ 18, Yang & Chen+ 18, Conneau & Lample 19, Ren & Wu+ 19], the

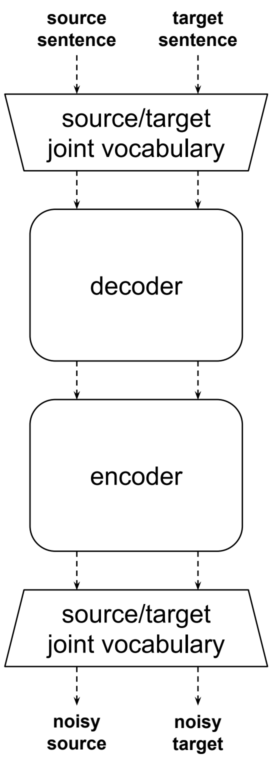


Figure 6.8: Denoising autoencoder training for source or target language.

whole monolingual data [Lample & Denoyer+ 18, Lample & Ott+ 18, Sun & Wang+ 19], or some size in between [Artetxe & Labaka+ 19, Ren & Zhang+ 19].

Initialization. To kickstart the iterative training, the model should be able to generate meaningful translations even in the first iteration. We cannot expect the training to progress from a randomly initialized network and the synthetic data generated by it.

Cross-lingual embeddings give a good starting point for the model by defining a joint continuous space shared by multiple languages, as discussed in Section 6.3.1. It can be learned either at word level [Artetxe & Labaka+ 17, Conneau & Lample+ 18] or at sentence level [Conneau & Lample 19] using only monolingual corpora. On the other hand, the whole encoder/decoder parameters can also be initialized with cross-lingual sequence training [Conneau & Lample 19, Ren & Wu+ 19, Song & Tan+ 19]. In addition to initializing the parameters, we may use word-by-word back-translations in the first epoch of the iterative training (Section 6.4.2), which are generated by cross-lingual word embeddings.

Cross-lingual word embedding has limited performance among distant languages [Søgaard & Ruder+ 18, Nakashole & Flauger 18] and so does cross-lingual LM [Pires & Schlinger+ 19]. Section 6.4.3 shows the impact of a poor initialization.

Denoising Autoencoder. Initializing the word embedding layers enables the model to perform cross-lingual matching in the lexical embedding space, but does not provide any information on word orders or generation of text. Cross-lingual LMs encode word sequences in different languages, but they are not explicitly trained to reorder source words to the target language syntax. Neither way initializes the crucial parameters for reordering: the encoder-decoder attention and the recurrence on decoder states. As a result, an initial model for unsupervised NMT tends to generate word-by-word translations with little reordering, which are non-fluent when source and target languages have distinct word orders. Training on such data discourages the model from reordering words, which might cause a vicious cycle by generating even less-reordered synthetic sentence pairs in the next iterations.

Accordingly, unsupervised NMT employs an additional training objective of denoising autoencoding (Section 6.3.3). Given a clean sentence, artificial noises are injected, e.g. deletion or permutation of words, to make a corrupted input. The denoising objective trains the model to reorder the noisy input to the correct syntax, which is essential for generating fluent outputs. This is done for each language individually with monolingual data, as shown in Figure 6.8. Once the model is sufficiently trained for denoising, we suggest removing the objective or reducing its weight. At the later stages of training, the model gets improved in reordering and translates better; learning to denoise might hurt the performance in clean test sets.

Adversarial Training. In addition, we may think of employing adversarial training to make the model more cross-lingual, helping the knowledge transfer between the two tasks. Specifically, we apply an adversarial loss to the encoder outputs, making it produce language-independent representations even though fed with both source and target sentences [Lample & Denoyer+ 18]. We argue that this potentially avoids the problem of input copying, where the decoder can distinguish the language from encoder representations and be encouraged to output the same language. The loss formulation is similar to the word embedding case (Equation 6.31-6.32):

$$\mathrm{L}_{\text{dis}}(\theta_{\text{dis}}) = -\frac{1}{|\mathcal{C}_f|}\sum_{f_1^J\in\mathcal{C}_f}\log p_{\theta_{\text{dis}}}\big(1|\mathrm{E}(f_1^J)\big) - \frac{1}{|\mathcal{C}_e|}\sum_{e_1^I\in\mathcal{C}_e}\log p_{\theta_{\text{dis}}}\big(0|\mathrm{E}(e_1^I)\big) \tag{6.36}$$

$$\mathrm{L}_{\text{gen}}(\theta) = -\frac{1}{|\mathcal{C}_f|}\sum_{f_1^J\in\mathcal{C}_f}\log p_{\theta_{\text{dis}}}\big(0|\mathrm{E}_\theta(f_1^J)\big) - \frac{1}{|\mathcal{C}_e|}\sum_{e_1^I\in\mathcal{C}_e}\log p_{\theta_{\text{dis}}}\big(1|\mathrm{E}_\theta(e_1^I)\big) \tag{6.37}$$

where the encoder is trained to fool a discriminator that attempts to detect the language of the input sentence after the encoder module. The discriminator is modeled position-wise with a simple feedforward binary classifier, i.e. $p_{\theta_{\text{dis}}}(\zeta|\mathbf{h}_1^J) = \prod_{j=1}^{J} p_{\theta_{\text{dis}}}(\zeta|\mathbf{h}_j)$.

6.4.3 Experiments

Our experiments on the unsupervised sequence-to-sequence NMT consist of two parts. Firstly, we analyze the impact of each component in the training process, optimizing the performance and obtaining a better understanding of the method. Secondly, given the most optimized recipe in our resources, we test its performance on various tasks and data settings to reveal the condition under which the unsupervised learning is indeed useful in practice.

Analysis of Components

The first set of experiments were done on the WMT 2018 German→English task, where we did not use any of the provided bilingual data but exploited monolingual data only. We used the full monolingual corpora (100M sentences on each side) of Section B.2 for training the initial word embeddings and sampled 5M sentences on each side for the translation model training. Since we learned from Section 6.3.4 that subword-level cross-lingual embedding is less effective than the word level, the training corpora were segmented into words. We kept only the top 50k frequent words and replaced all other words with `<unk>` tokens. In training, we only used sentences up to 50 words in length. Furthermore, to ease the cross-lingual mapping, we applied lowercasing and replaced all occurrences of numbers with a special category token. After decoding, the category tokens in the output were reverted to the corresponding numbers by monotonically matching them with the original source sentence. The `<unk>` tokens were replaced by real words in the same manner. Also, casing was recovered from the lowercased output with a frequent caser.

For efficiency reasons, the encoder and decoder both have 4 layers, where the embedding/hidden layer size is 300 with the feedforward sublayer having 2048 nodes. Monolingual word embeddings were pre-trained using skip-gram of FASTTEXT [Bojanowski & Grave+ 17] only for words that appear at least 10 times in the training data. Cross-lingual embedding mapping was trained with MUSE [Conneau & Lample+ 18] for 10 epochs of the adversarial setting and 10 steps of the Procrustes refinement procedure. The iterative back-translation was done at corpus level, where the back-translations were newly generated for each epoch. In the initial iteration, training examples were generated word-by-word by the pre-trained cross-lingual word embedding instead of the sequence-to-sequence model, which consistently showed more stable results. Each component of the loss function was equally weighted and combined for every batch instead of alternating it for each batch. We implemented the whole unsupervised learning framework based on SOCKEYE [Hieber & Domhan+ 17]. The training was done for five epochs (~800k batches) with batch size of 32 sentences, which amounts to rougly 6 days with a single GPU.

Sharing of Model Components. First, we inspect the effect of bidirectional parameter sharing (Section 6.4.1) which is a distinguishing property in modeling compared to normal supervised NMT. Table 6.12 shows unsupervised translation performance with different model components for the sharing. We compare shared and unshared variants of the embedding layers (along with the vocabularies) and the decoder, where the encoder was always shared. When sharing the vocabularies, we limit the resulting joint vocabulary to the top 50k most common words in both languages. This leads to an improvement of up to 0.9% BLEU and 0.7% TER better than using separate vocabularies. The best results are obtained by sharing all components, which maximizes the knowledge transfer between source and target languages.

Table 6.12: Effect of model sharing in unsupervised sequence-to-sequence NMT (WMT 2018 German→English).

		`newstest2017`		`newstest2018`	
Embedding Layers	Decoder	BLEU [%]	TER [%]	BLEU [%]	TER [%]
Shared	Shared	**14.9**	**72.7**	**18.1**	**67.1**
	Unshared	14.3	73.3	17.3	68.0
Unshared	Shared	14.5	73.3	17.2	67.8
	Unshared	13.8	76.2	16.1	72.0

Initialization of Embeddings. The basic training philosophy (iterative generation of hypotheses that are used to update the model) was already developed in decipherment works (Section 6.2.2); in the case of a sequence-to-sequence NMT, the initialization is considered to be crucial since the model is so complicated that starting from random parameters is too difficult. To confirm this, we compare different initialization schemes of embeddings in Table 6.13. Note that, if it is not cross-lingual, the first back-translations were generated by the NMT model itself. Without any prior knowledge (random initialization), the model fails to produce proper translations. Monolingual embedding slightly improves this, but it is still poor: below 10% BLEU. Once the embeddings are more language-independent (cross-lingual), the model is able to achieve much better metric scores.

This supports the argument that it is better to learn language-independent representations in unsupervised NMT. The model should be shared between source and target languages for effective training (Table 6.12). Also, the same model is used to translate any input language to any output language: source→target and target→source (iterative back-translation), source→source and target→target (denoising autoencoder). Cross-lingual embedding facilitates the diverse roles of the model by promoting language-level abstraction in the representation space, while monolingual embeddings have fundamentally differing word distributions. On a side note, freezing the pre-trained embeddings during iterative back-translation is detrimental to translation quality. For the unsupervised NMT, it is beneficial to tune the entire parameters to have the optimal results, including the embedding layers.

Table 6.13: Comparison of embedding layer initialization in unsupervised sequence-to-sequence NMT (WMT 2018 German→English).

	`newstest2017`		`newstest2018`	
Embedding Initialization	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
Random	4.4	93.7	4.3	92.5
Monolingual embedding	7.6	88.0	8.3	85.5
Cross-lingual embedding	**14.9**	**72.7**	**18.1**	**67.1**
+ Freeze embedding	14.0	75.8	16.9	71.5

Training Objectives. In Table 6.14, we provide an ablation study of multiple training objectives for unsupervised sequence-to-sequence NMT. Translation objective alone can reach a performance above 10% Bleu, yet 3-4% Bleu worse without the denoising autoencoder. This agrees with the finding of [Lample & Denoyer+ 18], which also shows that the performance drops significantly when removing the autoencoder objective. We observe that the outputs from an autoencoder-free system barely have a reordering operation and largely resemble word-by-word translation, on which it is trained in the first iteration. The autoencoding objective teaches the model to reorder, but it is not necessary in the later stage of training, as shown in the third row. Dropping it even gives a small additional improvement around +0.3% Bleu and reduces the training time. The adversarial losses significantly hurt the performance, which contradicts the results of [Lample & Denoyer+ 18]. We argue that it is difficult to find a balanced curriculum where both the position-independent discriminator and the sequence-level NMT model grow stronger at the same pace to interact effectively each other.

Table 6.14: Effect of training objectives for unsupervised sequence-to-sequence NMT (WMT 2018 German→English).

	`newstest2017`		`newstest2018`	
Training Objective	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
Translation	11.9	85.0	14.2	76.1
+ Denoising autoencoder	14.9	72.7	18.1	67.1
+ No denoising autoencoder from 4th iteration	**15.2**	**72.6**	**18.3**	**66.9**
+ Adversarial training	13.9	74.2	16.9	69.0

After optimizing the factors above, the final model consists of a word-based model with separate vocabularies, trained with the batch optimization method, initialized with cross-lingual embeddings, applies embedding weight normalization and is trained with a learning rate of $3 \cdot 10^{-4}$. We also built an ensemble system consisting of 4 variations of the single-best model, varying in learning rate values ($3 \cdot 10^{-4} \rightarrow 10^{-4}$), feed-forward projection hidden sizes ($2048 \rightarrow 1024$) and monolingual, instead of cross-lingual, embedding initialization; this was submitted to the WMT 2018 unsupervised news translation task and ranked top in the evaluation. Table 6.15 has an excerpt of the translation outputs from the winning systems.

Table 6.15: Example translation outputs for `newstest2018` from the winning system of WMT 2018 unsupervised news translation tasks [Graça & Kim+ 18].

de-en		Bleu [%]
Source	Die Stadt hat sich in alle Richtungen ausgedehnt und ist zusammengewachsen.	18.6
Hypothesis	The city has stretched in all directions and is cohesive.	
Reference	The city has stretched out in all directions and coalesced.	

en-de		Bleu [%]
Source	A crypt chapel, where they are now digging tunnels for the S-Bahn.	14.8
Hypothesis	Eine Gruft Kapelle, wo Sie nun Tunnel für die S-Bahn wühlen.	
Reference	Eine Gruftkapelle, wo nun für den S-Bahn-Tunnel gegraben wird.	

Usefulness in Practice

We have seen that sequence-level unsupervised training for translation functions to a certain level of quality, e.g. above 15% Bleu on the test sets in the same domain as the training data (news) in German→English. However, apart from the academic discovery, we are also interested in actually applying the unsupervised NMT to a real-world task that achieves a poor quality with traditional supervised or semi-supervised NMT. The following series of experiments reveal the conditions under which the unsupervised sequence-to-sequence MT succeeds or fails to produce meaningful translations, and accordingly whether it is indeed useful in practical data settings. For a comprehensive and pragmatic study on the performance of unsupervised NMT in practice, we test the method in ten translation tasks in the following five language pairs:

- WMT 2018 German→English: similar languages, abundant bilingual/monolingual data (Section B.2)
- WMT 2018 Russian→English: distant languages, abundant bilingual/monolingual data, similar sizes of the alphabet (Section B.14)
- WMT 2019 Chinese→English: distant languages, abundant bilingual/monolingual data, very different sizes of the alphabet (Section B.15)
- WMT 2019 Kazakh→English: distant languages, scarce bilingual data, abundant monolingual data (Section B.16)
- WMT 2019 Gujarati→English: distant languages, scarce bilingual/monolingual data (Section B.17)

For each task, we compare the unsupervised performance with its supervised and semi-supervised counterparts. In addition, we make the monolingual training data vary in size and domain to cover many more scenarios, showing under which conditions unsupervised NMT works poorly. The datasets were preprocessed as in Section 3.6.1, where we additionally used the jieba segmenter[1] for Chinese.

For unsupervised experiments, we ran the Xlm toolkit[2] by [Conneau & Lample 19], which implements the model described in Section 6.4.1. It shares the whole model parameters between the source and target languages. The back-translations were done with greedy search for each mini-batch of 16k tokens. The weight of the denoising objective started with 1 and linearly decreased to 0.1 until 100k updates, and then decreased to 0 until 300k updates. The model's encoder and decoder were both initialized with the same pre-trained cross-lingual LM. We removed the language

[1]https://github.com/fxsjy/jieba
[2]https://github.com/facebookresearch/XLM

Table 6.16: Comparison among supervised, semi-supervised, and unsupervised learning.

(a) WMT 2018 German→English

Data Size (#sents)				`newstest2017`		`newstest2018`	
de-en	de	en	Method	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
5.9M	-	-	Supervised	33.3	53.2	39.5	45.4
5.9M	-	10M	Semi-supervised	**35.8**	**50.7**	**43.6**	**41.6**
-	100M	100M	Unsupervised	20.5	66.8	23.8	61.4

(b) WMT 2018 Russian→English

Data Size (#sents)				`newstest2017`		`newstest2018`	
ru-en	ru	en	Method	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
25M	-	-	Supervised	33.5	52.2	29.1	58.2
25M	-	10M	Semi-supervised	**36.5**	**48.4**	**30.8**	**55.4**
-	72M	72M	Unsupervised	14.7	73.6	12.0	78.1

(c) WMT 2019 Chinese→English

Data Size (#sents)				`newsdev2019`		`newstest2019`	
zh-en	zh	en	Method	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
19M	-	-	Supervised	22.2	67.1	24.1	64.7
19M	-	10M	Semi-supervised	**23.4**	**65.7**	**24.9**	**63.9**
-	31M	31M	Unsupervised	1.7	94.3	1.8	93.9

(d) WMT 2019 Kazakh→English

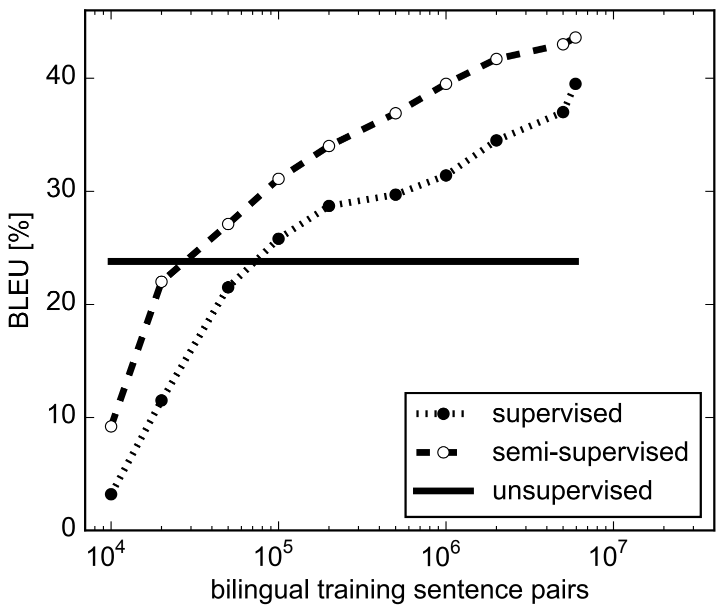
Data Size (#sents)				`newsdev2019`		`newstest2019`	
kk-en	kk	en	Method	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
222k	-	-	Supervised	10.3	84.3	10.3	83.8
222k	-	4M	Semi-supervised	**13.7**	**80.9**	**12.5**	**82.2**
-	19M	19M	Unsupervised	1.1	102.2	2.0	99.8

(e) WMT 2019 Gujarati→English

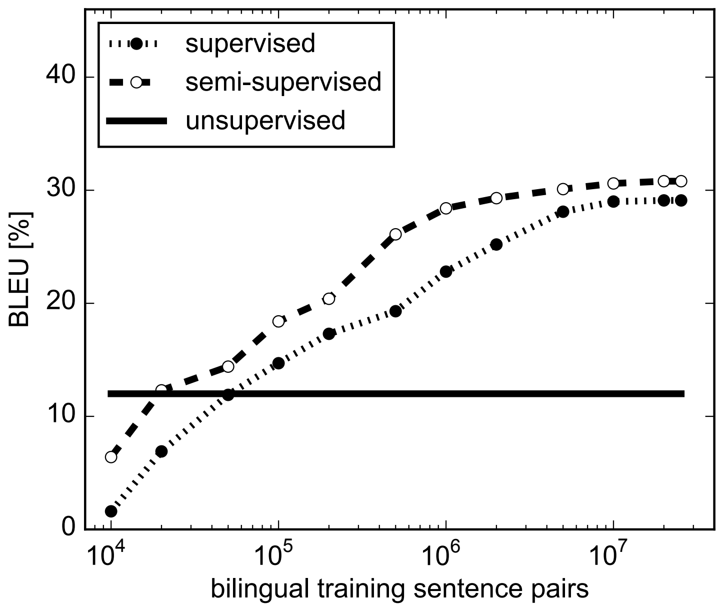
Data Size (#sents)				`newsdev2019`		`newstest2019`	
gu-en	gu	en	Method	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
156k	-	-	Supervised	9.8	79.4	9.9	80.7
156k	-	4M	Semi-supervised	**20.5**	**67.4**	**14.2**	**75.0**
-	4.1M	4.1M	Unsupervised	1.1	140.5	0.6	130.6

embeddings from the encoder for better cross-lingual abstraction. Unless otherwise specified, we used the same monolingual training data for both pre-training and translation training. For the pre-training, we set the batch size to 256 sentences (around 66k tokens). Training was done with a checkpoint frequency of 200k sentences, and we stopped the training when the validation perplexity (pre-training) or Bleu (translation training) was not improved for ten checkpoints. We extensively tuned the hyperparameters for a single GPU with 12GB memory, which is widely applicable to moderate industrial/academic environments. All other hyperparameter values follow the recommended settings of Xlm.

Unsupervised vs. (Semi-)Supervised. We first address the most general question of this chapter: For NMT, can unsupervised learning replace semi-supervised or supervised learning?



(a) German→English



(b) Russian→English

Figure 6.9: Supervised and semi-supervised learning over bilingual training data size. Unsupervised learning (horizontal line) uses all monolingual data available (see Section B.2 and B.14).

Table 6.16 compares the unsupervised performance to simple supervised and semi-supervised baselines. Supervised experiments used the same hyperparameters as the unsupervised learning, except for 12k tokens for the batch size, 0.0002 for the initial learning rate, and 10k batches for each checkpoint. If the bilingual training data contains less than 500k sentence pairs, we reduced the BPE merges to 8k, the batch size to 2k, and the checkpoint frequency to 4k batches; we also increased the dropout rate to 0.3 [Sennrich & Zhang 19]. Semi-supervised experiments continued the training from the supervised baseline with back-translations added to the training data. We used 4M back-translated sentences for the low-resource cases, i.e. if the original bilingual data has less than 500k lines, and 10M back-translated sentences otherwise.

In all tasks, unsupervised learning shows much worse performance than (semi-)supervised learning. It produces readable translations in two high-resource language pairs (German→English and Russian→English), but their scores are only around half of the semi-supervised systems. In the other three language pairs, unsupervised NMT fails to converge at any meaningful optimum, reaching less than 3% Bleu scores. Note that, in these three tasks, source and target languages are very different in the alphabet, morphology, and word order, etc. The results in Kazakh→English and Gujarati→English show that the current unsupervised NMT cannot be an alternative to (semi-)supervised NMT in low-resource conditions.

To discover the precise condition where the unsupervised learning is useful in practice, we vary the size of the given bilingual training data for (semi-)supervised learning and plot the results in Figure 6.9. Once we have 50k bilingual sentence pairs in German→English, simple semi-supervised learning already outperforms unsupervised learning with 100M monolingual sentences in each language. Even without back-translations (supervised), 100k-sentence bilingual data is sufficient to surpass unsupervised NMT. In the Russian→English task, the unsupervised learning performance can be more easily achieved with only 20k bilingual sentence pairs using semi-supervised learning. This might be due to the fact that Russian and English are more distant from each other than German and English, thus the bilingual training signal is more crucial for Russian→English. Note that for these two language pairs, the bilingual data for supervised learning are from many different text domains, whereas the monolingual data are from exactly the same domain of the test sets. Even with such an advantage, the large-scale unsupervised NMT cannot compete with supervised NMT with tiny out-of-domain bilingual data.

Monolingual Data Size. In this section, we analyze how much monolingual data is necessary to make unsupervised NMT produce a reasonable performance. Figure 6.10 shows the unsupervised results with different amounts of monolingual training data. We keep the source

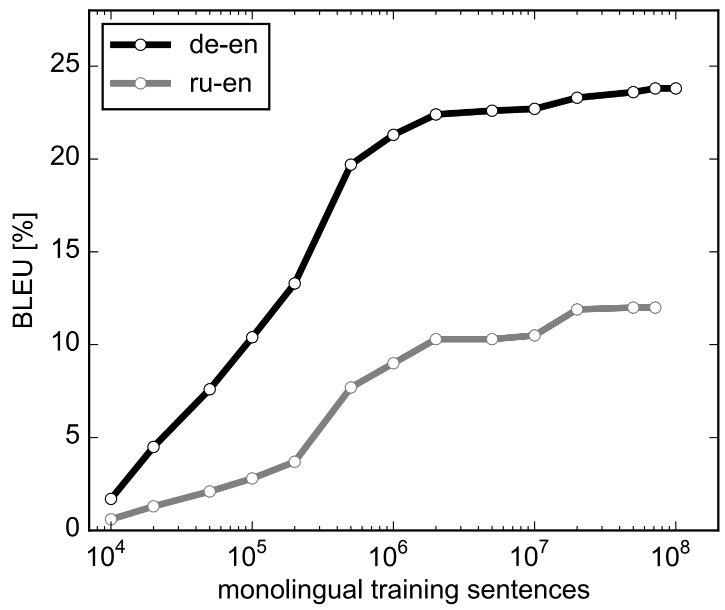


Figure 6.10: Unsupervised NMT performance over the size of monolingual training data, where source and target sides are the same size.

and target data the same size, and the domain is also the same for both (web-crawled news). For German→English, training with only 1M sentences already gives a reasonable performance, which is only around 2% Bleu behind the 100M-sentence case. The performance starts to saturate after 5M sentences, with only marginal improvements from using more than 20M sentences. We observe a similar trend in Russian→English. This shows that, for the performance of unsupervised NMT, using a massive amount of monolingual data is not as important as the similarity of source and target languages. Compared to supervised learning (see Figure 6.9), the performance saturates faster when increasing the training data, given the same model size.

Unbalanced Data Size. What if the size of the available monolingual data is largely different for source and target languages? This is often the case for low-resource language pairs involving English, where there is plenty of data for English but not for the other side. Our experiments so far intentionally use the same number of sentences for both sides. In Figure 6.11, we reduced the source data gradually while keeping the large target data fixed. To counteract the data imbalance, we oversampled the smaller side to make the ratio of source-target 1:1 for BPE learning and mini-batch construction [Conneau & Lample 19]. We compare such unbalanced data settings to the previous equal-sized source/target settings. Interestingly, when we decrease the target data accordingly (balanced, solid line), the performance is similar or sometimes better than using the full target data (unbalanced, dashed line). This means that it is not beneficial to use oversized data on one side in unsupervised NMT training. If the data is severely unbalanced, the distribution of the smaller side should be much sparser than that of the larger side. The network tries to generalize more on the smaller data, reserving the model capacity for smoothing [Olson & Wyner+ 18]. Thus it learns to represent a very different distribution for each side, which is challenging in a shared model (Section 6.4.1). This could be why there is no merit in using larger data on one side.

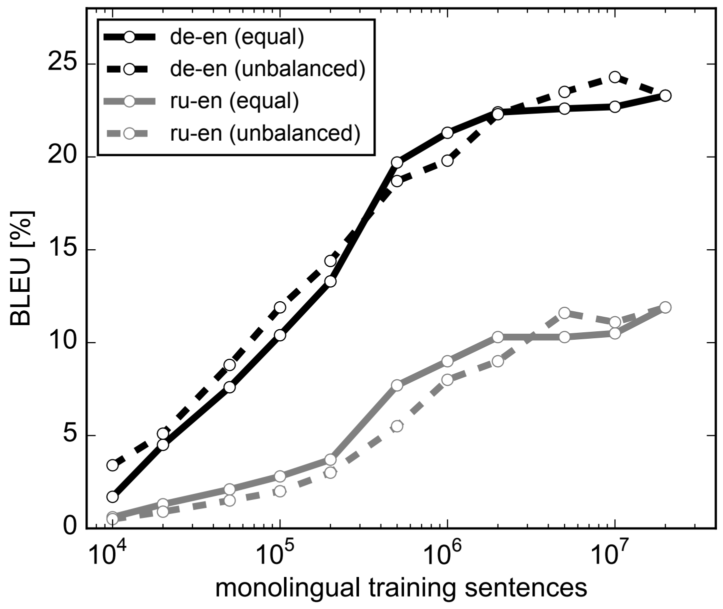


Figure 6.11: Unsupervised NMT performance over source training data size, where the target training data is fixed to 20M sentences (dashed line). Solid line is the case where the target data has the same number of sentences as the source side.

Domain Similarity. In high-resource language pairs, it is feasible to collect monolingual data from the same domain on both source and target languages. However, for low-resource

Table 6.17: Unsupervised NMT performance where source and target training data are from different domains. The data size on both sides is the same (20M sentences).

		WMT 2018 German→English				WMT 2018 Russian→English			
		`newstest2017`		`newstest2018`		`newstest2017`		`newstest2018`	
Target Domain	Source Domain	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
Newswire	Newswire	19.9	67.4	23.3	61.8	14.7	73.1	11.9	78.0
	Politics	**9.8**	**90.2**	**11.5**	**86.1**	**3.1**	**118.7**	**2.3**	**124.9**
	Random	15.1	72.0	18.4	66.6	8.5	81.0	6.9	84.8

Table 6.18: Unsupervised NMT performance where source and target training data are from the same domain (newswire) but different years (WMT 2019 Chinese→English).

Years of News (Target)	Years of News (Source)	Data Size (#sents)	`newsdev2017`		`newstest2018`	
			Bleu [%]	Ter [%]	Bleu [%]	Ter [%]
2014-2017	2008-2018	1.7M	5.8	88.6	5.7	87.9
	1995-2008	28.6M	**1.7**	**93.0**	**1.7**	**93.3**

language pairs, it is difficult to match the data domain of both sides on a large scale. For example, our monolingual data for Kazakh is mostly from Wikipedia and Common Crawl, while the English data is solely from News Crawl. In this section, we study how the domain similarity of monolingual data on the two sides affects the performance of unsupervised NMT. In Table 6.17, we artificially change the domain of the source side to politics (UN Corpus[3]) or random (Common Crawl), while keeping the target domain fixed to newswire (News Crawl). The results show that domain matching is critical for unsupervised NMT. For instance, although German and English are very similar languages, we see the performance of German→English deteriorate by -11.8% Bleu due to the domain mismatch.

Table 6.18 shows a more delicate case where we keep the same domain for both sides (newswire) but change the providers and years of the news articles. Our monolingual data for Chinese (Section B.15) consist mainly of News Crawl (from years 2008-2018) and Gigaword 4th edition (from years 1995-2008). We split out the News Crawl part (1.7M sentences) and trained an unsupervised NMT model with the same amount of English monolingual data (from News Crawl 2014-2017). Surprisingly, this experiment yields much better results than using all available data. Even if the size is small, the source and target data are collected in the same way (web-crawling) from similar years (2010s), which seems to be crucial for unsupervised NMT to work.

On the other hand, when using the Gigaword part (28.6M sentences) on Chinese, unsupervised learning again does not function properly. Now the source and target text are from different decades; the distribution of topics might be different. Also, the Gigaword corpus is from traditional newspaper agencies, which can have a different tone from the online text of News Crawl. Despite the large scale, unsupervised NMT proves to be sensitive to a subtle discrepancy of topic, style, period, etc. between source and target data. These results agree with [Søgaard & Ruder$^+$ 18], who show that modern cross-lingual word embedding methods fail in domain mismatch scenarios.

Initialization vs. Translation Training. Thus far, we have seen a number of cases where unsupervised NMT breaks down. But which part of the learning algorithm is more responsible for the performance: initialization or translation training (see Section 6.4.2)? In Figure 6.12, we control the level of each of the two training stages and analyze its impact on the final performance. We

[3]https://conferences.unite.un.org/uncorpus

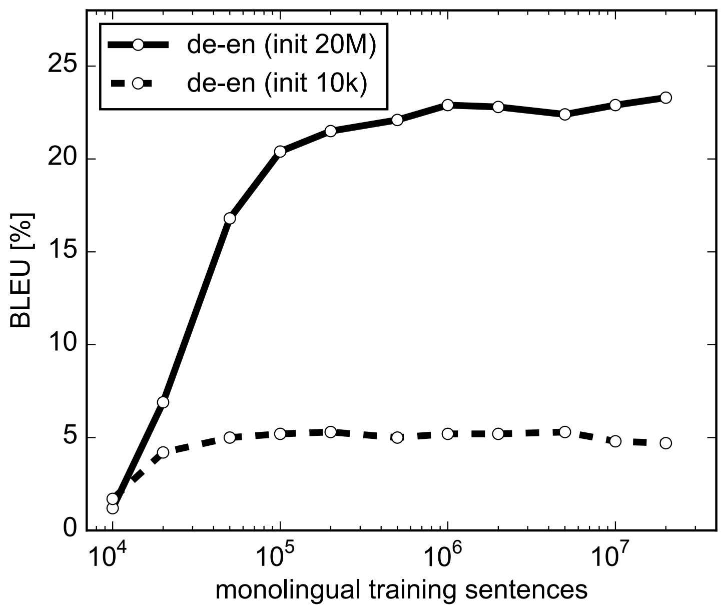


Figure 6.12: Unsupervised NMT performance over the training data size for translation training, where the pre-training data for initialization is fixed (10k or 20M sentences).

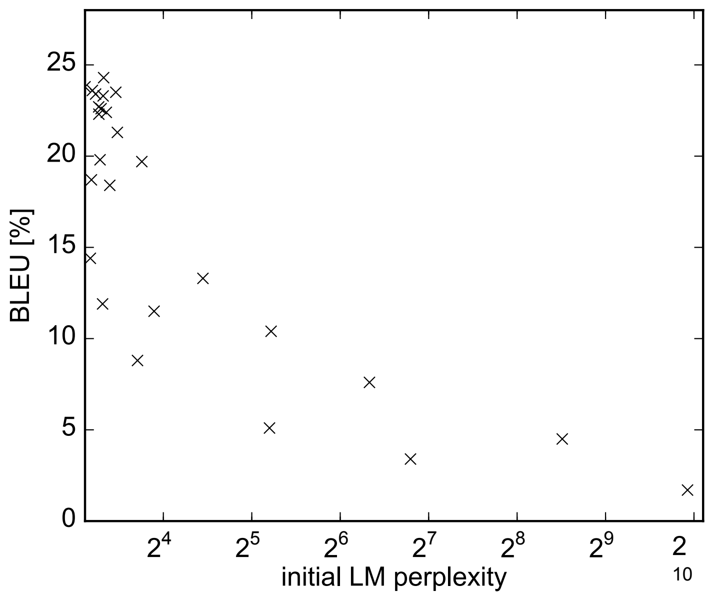


Figure 6.13: Unsupervised NMT performance over the validation perplexity of the initial cross-lingual LM (**de-en**).

pre-trained two cross-lingual LMs as initializations of different quality: bad (using 10k sentences) and good (using 20M sentences). For each initial point, we continued the translation training with different amounts of data from 10k to 20M sentences. From the bad initialization, unsupervised learning cannot build a reasonable NMT model, no matter how much data is used in translation training. When the initial model is strong, it is possible to reach 20% Bleu by translation training with only 100k sentences. Using 1M sentences in translation training, the performance is already comparable to its best. Once the model is pre-trained well for cross-lingual representations, fine-tuning the translation-specific components seems manageable with relatively small amounts of data.

This demonstrates the importance of initialization over translation training in the current unsupervised NMT. Translation training relies solely on model-generated inputs, i.e. back-translations, which do not reflect the true distribution of the input language when generated with a poor initial model. In Figure 6.13, we plot all of the German→English unsupervised results we conducted up to the previous section. It shows that the final performance generally correlates with the initialization quality.

Qualitative Examples. We also analyze the translation outputs of unsupervised systems to find out why they record such low Bleu scores. Do unsupervised systems have particular problems in the outputs other than limited adequacy/fluency? Table 6.19 shows translation examples from the unsupervised systems. The first notable problem is copying input words to the output. This happens when the encoder has poor cross-linguality, i.e. does not concurrently model two languages well in a shared space. The decoder then can easily detect the input language by reading the encoder and may emit output words in the same language.

A good cross-lingual encoder should not give away information on the input language to the decoder. The decoder must instead rely on the ouptut language embeddings or an indicator token (e.g. `<2en>`) to determine the language of output tokens. As a simple remedy, we removed the language embeddings from the encoder and obtained consistent improvements, e.g. from 4.3% to 11.9% Bleu in Russian→English. However, the problem still remains partly even in our best-performing unsupervised system (the first example). The copying occurs more often in inferior systems (the last example), where the poor initial cross-lingual LM is the main reason for the worse performance (Figure 6.13). Note that the autoencoding objective also encourages the model

Table 6.19: Problematic translation outputs from unsupervised NMT systems (input copying, *ambiguity in the same context*).

de-en		Bleu [%]
Source	Seit der ersten Besichtigung wurde die *1.000 Quadratfuß* große ...	23.8
Hypothesis	Since the first Besichtigung, the *3,000 square* fueled ...	
Reference	Since the first viewing, the 1,000sq ft flat has ...	

de-en		Bleu [%]
Source	*München* 1856: *Vier* Karten, die Ihren Blick auf die *Stadt* verändern	10.4
Hypothesis	*Australia* 1856: *Eight* things that can keep your way to the *UK*	
Reference	Munich 1856: Four maps that will change your view of the city	

ru-en		Bleu [%]
Source	В ходе первоочередных оперативно-следственных мероприятий ус-тановлена личность роженицы	12.0
Hypothesis	The первоочередных оперативно-следственных мероприятий have been established by the dolphin	
Reference	The identity of the mother was determined during preliminary investigative and operational measures	

zh-en		Bleu [%]
Source	... 调整要兼顾生产需要和消费需求。	1.5
Hypothesis	... 调整要兼顾生产需要 and 消费需求.	
Reference	... adjustment must balance production needs with consumer demands.	

to generate outputs in the input language.

Another problem is that the model cannot distinguish words that appear in the same context. In the second example, the model knows that *Vier* in German (*Four* in English) is a number, but it generates the wrong number in English (*Eight*). The initial LM is trained to predict either *Four* or *Eight* given the same surrounding words (e.g. 1856, things) and has no clue to map *Four* to *Vier*. The model cannot learn these mappings by itself with back-translations. This problem can be partly solved by subword modeling [Bojanowski & Grave$^+$ 17] or orthographic features [Riley & Gildea 18, Artetxe & Labaka$^+$ 19], which are, however, not effective for language pairs with disjoint alphabets.

6.5 Comparison among the Presented Methods and Other Work

We compare the best produced results of this thesis to other literature in Table 6.20. The first row is the result of neural cascaded combination (Section 6.3), which shows considerable performance for unsupervised learning but clearly falls behind the second row using a sequence-to-sequence model (Section 6.4). This means that, when a proper strategy for end-to-end training is used, more expressive modeling is better for unsupervised MT. The last two rows also use sequence-to-sequence NMT; despite using the same toolkit, [Conneau & Lample 19] show superior performance to this thesis. This difference is attributed to their using more data and correspondingly a much larger batch size (12k vs. 1M tokens). Such a large mini-batch can only be used with dozens of high-end GPUs; however, all results of this thesis were produced with only one GPU. This comparison highlights the great importance of large-scale training in unsupervised NMT.

Table 6.20: Comparison to other work on unsupervised NMT (WMT 2018 German→English).

Data Size (#sents)			`newstest2016`
			BLEU
de	en	Work	[%]
100M	100M	This thesis (neural cascaded combination)	17.2
		This thesis (using XLM)	**23.3**
261M	193M	4-layer Transformer [Lample & Ott$^+$ 18]	21.0
		XLM [Conneau & Lample 19]	34.3

6.6 Conclusion and Contributions

This chapter demonstrates the chronological development of unsupervised methods for MT, from a primitive combination of count-based models to an end-to-end neural model. To start with, we extended a classical word-based decipherment framework for realistic translation scenarios, which had worked only for a few dozens of cipher characters. With sparse table and word class initialization, we improved accuracy from 70.2% to 76.2% on the EUTRANS Spanish→English task without worrying about memory footprint and random restarts. We also investigated the usage of neural network lexicons in the decipherment modeling. We derived the stochastic training algorithm for the neural lexicons and verified the advantage of discriminative modeling, which marked 73.3% accuracy on the EUTRANS task, which is better than the baseline but worse than our optimized sparse tables. We did not observe a performance gain by including context words in the lexicons due to the increased difficulty of training.

Embracing the recent development of neural word/text modeling, we proposed a cascaded combination of the neural components as an alternative. Combining cross-lingual word embedding and LM followed by a sequence-level denoiser, we achieved e.g. 17.2% BLEU on WMT

German→English `newstest2016`. This method is efficient in training in the sense that it does not need an iterative generation of intermediate hypotheses. For training the cross-lingual embedding, which is the foremost part of this combination, we compared corpus- and vocabulary-based methods and different training criteria, leading to the conclusion that the best performance can be obtained by vocabulary-based adversarial training followed by minimum squared error optimization. In addition, we proved that a phrase-level embedding outperforms its word-based counterpart, but underperforms in the cascaded combination. Within the denoising process, we devised a novel insertion noise which shows a promising performance even combined with other noise types.

Lastly, we investigated the end-to-end unsupervised training of a sequence-to-sequence NMT model. We empirically found that the following factors are important to stabilize and optimize the training process: 1) the full model should be shared between source and target languages, 2) the model should be initialized cross-lingually, and 3) a denoising autoencoder should be combined with the translation objectives but can be omitted in the later stage of training. The method was thoroughly evaluated in numerous real and artificial translation tasks, showing that the performance is seriously degraded when linguistic or domain similarity between source and target languages is low. Our experiments question the usefulness of the method in practice; a simple, non-tuned semi-supervised baseline trained with less than 50k bilingual sentence pairs is sufficient to outperform our best large-scale unsupervised results. Our analysis found that poor initialization of the model is the most critical factor for the unsupervised method. The individual contributions and achievements from this chapter are listed below.

Word-based Decipherment. Sparse table and word class initialization of Section 6.2.2 were devised, implemented, and experimentally evaluated by Yunsu Kim. Julian Schamper contributed to scientific elaboration of these methods, resulting in the publication at EACL 2017 [Kim & Schamper$^+$ 17]. This also reports the first acceptable result of unsupervised MT on a translation task with ~100k vocabulary words: 54.2% accuracy on the Europarl Spanish→English decipherment task (Section B.13). Neural network lexicon training was derived for decipherment by Yunsu Kim and implemented by Jiahui Geng and Yunsu Kim.

Neural Cascaded Model Combination. Yunsu Kim designed the unsupervised cascaded combination of neural components (Section 6.3). The LM integration into cross-lingual word embedding[4] was implemented by Jiahui Geng and the text sequence denoiser[5] was implemented by Yunsu Kim. The codes for both were refactored and made publicly available by Yunsu Kim. Experiments were conducted by Yunsu Kim and Jiahui Geng, which were published at EMNLP 2018 [Kim & Geng$^+$ 18]. Yunsu Kim devised the corpus-based and phrase-level learning algorithms for cross-lingual embedding, which were programmed and evaluated by Jiahui Geng. All contents of Section 6.3 were also reported in the Master's thesis of Jiahui Geng [Geng 18].

Sequence-to-Sequence Model. Miguel Graça implemented the sequence-to-sequence unsupervised NMT in Sockeye and led the component analysis part of Section 6.4.3. The resulting system ranked top among two participants in the WMT 2018 unsupervised German→English and English→German news translation tasks [Graça & Kim$^+$ 18]. For the task submission, the datasets were prepared and preprocessed by Yunsu Kim, the word-by-word translations were provided by Jiahui Geng, and the research discussions were supervised by Yunsu Kim and Julian Schamper. All experiments in the second part (practical and conditional evaluations) of Section 6.4.3 were designed and conducted by Yunsu Kim. The results were published at EAMT 2020 [Kim & Graça$^+$ 20]. Miguel Graça supported the experiments by automating the pipeline of running

[4]`https://github.com/yunsukim86/wbw-lm`
[5]`https://github.com/yunsukim86/sockeye-noise`

the XLM toolkit.

7. Scientific Achievements

In this thesis work, we investigated three categories of learning methods for low-resource NMT. In the following, we list the scientific achievements that are in line with the goals of Chapter 2.

- **Utilizing monolingual corpora**: We verified the optimal setting for the log-linear integration of an LM into an NMT model by studying its hyperparameters and comparing between sequence- and position-level combinations (Table 4.3). For generation of synthetic bilingual corpora, we proposed to restrict the sample space for synthetic sentences by a probability threshold or N-best list. We proved that the sampling with restriction provides a reasonable variety in the synthetic data and is the best method for the empirical performance (Figure 4.4). We suggested pre-training an NMT model with language model or Cloze task objectives, which proves to be effective in low-resource NMT (Figure 4.7). Finally, we drew a conclusion that synthetic data generation is the best practice for utilizing monolingual corpora in NMT (Table 4.12).
- **Utilizing bilingual corpora of other language pairs**: We proposed effective techniques to transfer a pre-trained NMT model to a low-resource language pair. For translation to a common target language, e.g. English, we proposed to relieve the vocabulary mismatch by using cross-lingual word embedding, train a more language-agnostic encoder by injecting artificial noises, and generate synthetic data easily from the pre-training data without back-translation. We showed that these methods outperform the multilingual model or semi-supervised baselines, while not requiring restructuring the vocabulary or retraining the model (Table 5.1). For non-English language pairs, we suggested pre-training NMT models with source-pivot and pivot-target parallel data and transfering the source encoder and the target decoder. To resolve the input/output discrepancy of the pre-trained encoder and decoder, we proposed to 1) consecutively pre-train the model for source$\rightarrow$pivot and pivot$\rightarrow$target, 2) append an additional layer after the source encoder which adapts the encoder output to the pivot language space, or 3) train a cross-lingual encoder over source and pivot languages. We proved that these methods are more effective than the multilingual model (Table 5.6) and also in a zero-shot scenario (Table 5.8). We also tested integrated training of cascaded pivoting models, showing that it improves the cascading but underperforms against the transfer methods (Table 5.13).
- **Training only with monolingual corpora**: We extended the word-base decipherment framework by filtering out unlikely lexicon entries according to the training progress (Table 6.1) and using word classes to learn a stable starting point for the training (Table 6.2). Also, we investigated usages of neural network lexicon structures which include source side context words, exploiting the generalized EM algorithm in training (Table 6.3). We proved that discriminative training is preferred for neural lexicon training, yet it underperforms compared to an optimized training of table lexicons (Table 6.5). We derived a novel cascaded combination of neural models from the log-linear combination, using cross-lingual word em-

bedding and a denoising autoencoder. This proved to be effective in unsupervised MT even without iterative generation of intermediate hypotheses (Table 6.7). Finally, we examined unsupervised training of sequence-to-sequence models. We proved that model sharing and cross-lingual initialization are important in optimizing the performance. In extensive experiments in various data settings, we found that the model initialization is very sensitive to the similarity of linguistic properties and domains between source and target languages. This leads to the conclusion that unsupervised NMT is not yet practical in many real-world low-resource scenarios where at least a handful of bilingual sentence pairs is given (Table 6.16).

A natural direction of future work would be combining knowledge in different chapters of this thesis, e.g. cross-lingual transfer (Chapter 5) to build a good inverse model which is used to generate high-quality synthetic data by restricted sampling afterwards (Chapter 4). We may test a variety of such combinations by differing the order and variants of our proposed techniques to find the optimal model building pipeline for each language pair.

As a side note, during the period of this thesis work, the author also contributed to other topics which are not directly related to the contents of this thesis. Here is the list of publications about such contributions:

- When and Why Is Document-level Context Useful in Neural Machine Translation? [Kim & Tran$^+$ 19]
- Learning Bilingual Sentence Embeddings via Autoencoding and Computing Similarities with a Multilayer Perceptron [Kim & Rosendahl$^+$ 19]
- The RWTH Aachen University Filtering System for the WMT 2018 Parallel Corpus Filtering Task [Rossenbach & Rosendahl$^+$ 18]
- A Comparative Study on Vocabulary Reduction for Phrase Table Smoothing [Kim & Guta$^+$ 16]

A. Symbols and Notations

Table A.1 puts together all symbols and notations used throughout this thesis. In general, a plain letter (e.g. x or X) denotes a scalar value, a boldface lowercase letter (e.g. $\mathbf{x}$) is a vector, a boldface uppercase letter (e.g. $\mathbf{X}$) is a matrix, and a calibrated uppercase letter (e.g. $\mathcal{X}$) denotes a set. A function is written with roman letters (e.g. L).

Table A.1: Symbols and notations in this thesis.

a	source-to-target word alignment	O	big-O notation
$\mathbf{b}$	bias vector	$\mathcal{O}$	set of orthogonal matrices
c	count	p	model distribution
$\mathcal{C}$	corpus	Pr	true distribution
d	dimension index	$\mathbf{q}$	query vector for attention
D	dimension	$\mathbf{Q}$	query matrix for attention
D	decoder	$\mathbf{s}$	decoder state vector
$\mathcal{D}$	dictionary	$\mathbf{S}$	decoder state matrix
e	target word	V	vocabulary size
E	encoder	$\mathbf{V}$	value matrix for attention
$\mathbf{E}$	target embedding matrix	$\mathcal{V}$	vocabulary
f	source word	$\mathbf{W}$	network weight matrix
$\mathbf{F}$	source embedding matrix	$\mathbf{x}$	similarity function operand
g	pivot word	$\mathbf{y}$	similarity function operand
H	LM observation histogram size	z	attention head index
$\mathbf{h}$	encoder representation vector	Z	number of attention heads
$\mathbf{H}$	encoder representation matrix	Z	normalization factor
i	target position index	α	attention weight
I	target sentence length	β	proxy orthogonality constant
j	source position index	γ	exponent for renormalization
J	source sentence length	δ	one-hot, discount coefficient
k	pivot position index	ϵ	label smoothing discount
K	pivot sentence length	λ	model component weight
$\mathbf{K}$	key matrix for attention	θ	model parameter
l	layer index	η	learning rate
L	number of layers	τ	probability threshold
L	loss function	ω	length penalty factor
m	model components index	ζ	language index (discriminator)
M	number of model components	$\tilde{}$	"updated"
n	order of grams	$\bar{}$	"modified"
N	beam size, N-best list size	$\hat{}$	"optimized"
$\mathbf{o}$	output logit	$'$	second running index

B. Corpora Statistics

In all corpora statistics, running words are counted after the tokenization step. The OOVs are based on the vocabulary defined by the bilingual training data, either at the word or subword level. The OOV rates are computed against the running words or subwords and rounded to one decimal place. Note that some training datasets are adjusted to the purpose of the experiments for this thesis, and thus the data sizes might be different than given in the original task description. For details of the data processing, see Section 3.6.1.

B.1 WMT 2017 Turkish-English News Translation Task

Table B.1: WMT 2017 Turkish↔English news translation task corpora statistics.

Name	Domain		Turkish	English
`train-bilingual`	news article	Sentence pairs	208k	
		Word vocabulary	173k	79k
		Running words	4.5M	5.1M
		Subword vocabulary	20100	
		Running subwords	5.6M	5.8M
`train-monolingual`	news article	Sentences	4.8M	100M
		Running words	76M	2.3B
		Running subwords	120M	3.0B
`newsdev2016`	news article	Sentence pairs	1001	
		Running words	17k	22k
		OOV words	1548 (9.4%)	815 (3.7%)
		Running subwords	25k	28k
		OOV subwords	1 (0.0%)	31 (0.1%)
`newstest2016`	news article	Sentence pairs	3000	
		Running words	52k	66k
		OOV words	4816 (9.2%)	3001 (4.5%)
		Running subwords	80k	87k
		OOV subwords	5 (0.0%)	32 (0.0%)
`newstest2017`	news article	Sentence pairs	3007	
		Running words	53k	68k
		OOV words	4672 (8.7%)	3225 (4.4%)
		Running subwords	81k	87k
		OOV subwords	9 (0.0%)	37 (0.0%)

- Original task: `http://www.statmt.org/wmt17/translation-task.html`
- Subword units: joint BPE with 20k merge operations, no vocabulary threshold

B.2 WMT 2018 German-English News Translation Task

Table B.2: WMT 2018 German↔English news translation task corpora statistics.

Name	Domain		German	English
`train-bilingual`	parliament speech, website crawl, news commentary, press release	Sentence pairs	5.9M	
		Word vocabulary	2.3M	1.1M
		Running words	137M	145M
		Subword vocabulary	52953	
		Running subwords	160M	157M
`train-monolingual`	news article	Sentences	100M	100M
		Running words	1.8B	2.3B
		Running subwords	2.3B	2.6B
`newstest2015`	news article	Sentence pairs	2169	
		Running words	44k	47k
		OOV words	1107 (2.5%)	551 (1.2%)
		Running subwords	55k	53k
		OOV subwords	0 (0.0%)	0 (0.0%)
`newstest2017`	news article	Sentence pairs	3004	
		Running words	64k	68k
		OOV words	2519 (3.9%)	1617 (2.4%)
		Running subwords	76k	73k
		OOV subwords	0 (0.0%)	0 (0.0%)
`newstest2018`	news article	Sentence pairs	2998	
		Running words	64k	68k
		OOV words	1604 (2.5%)	806 (1.2%)
		Running subwords	79k	77k
		OOV subwords	0 (0.0%)	0 (0.0%)

- Original task: `http://www.statmt.org/wmt18/translation-task.html`
- Subword units: joint BPE with 50k merge operations, vocabulary threshold 50

B.3 WMT 2016 Romanian-English News Translation Task

Table B.3: WMT 2016 Romanian↔English news translation task corpora statistics.

Name	Domain		Romanian	English
`train-bilingual`	news article	Sentence pairs	612k	
		Word vocabulary	131k	101k
		Running words	16M	16M
		Subword vocabulary	20301	
		Running subwords	18M	17M
`train-monolingual`	news article	Sentences	2.3M	55M
		Running words	55M	1.2B
		Running subwords	73M	1.6B
`newsdev2016`	news article	Sentence pairs	1999	
		Running words	52k	50k
		OOV words	1970 (%)	1678 (%)
		Running subwords	67k	63k
		OOV subwords	0 (0.0%)	22 (0.0%)
`newstest2016`	news article	Sentence pairs	1999	
		Running words	49k	48k
		OOV words	1528 (%)	1340 (%)
		Running subwords	63k	59k
		OOV subwords	4 (0.0%)	13 (0.0%)

- Original task: `http://www.statmt.org/wmt16/translation-task.html`
- Subword units: joint BPE with 20k merge operations, no vocabulary threshold

B.4 IWSLT 2018 Basque-English Low-Resource Translation Task

Table B.4: IWSLT 2018 Basque↔English low-resource translation task corpora statistics.

Name	Domain		Basque	English
`train-bilingual`	TED talk	Sentence pairs	5.6k	
		Word vocabulary	19k	10k
		Running words	98k	129k
		Subword vocabulary	10k	10k
		Running subwords	124k	130k
`train-monolingual`	wikipedia	Sentences	2.3M	-
		Running words	38M	-
		Running subwords	47M	-
`dev2018`	TED talk	Sentence pairs	1140	
		Running words	19k	26k
		OOV words	2867 (15%)	1728 (6.6%)
		Running subwords	25k	27k
		OOV subwords	861 (3.5%)	1749 (6.4%)
`tst2018`	TED talk	Sentence pairs	1051	
		Running words	17k	-
		OOV words	2515 (15%)	-
		Running subwords	22k	-
		OOV subwords	712 (3.2%)	-

- Original task: `https://sites.google.com/site/iwsltevaluation2018/TED-tasks`
- Subword units: separate BPE with 20k (source) or 50k (target) merge operations, no vocabulary threshold
- The target reference of `tst2018` is not publicly released and thus its statistics are not reported. The Bleu and Ter scores of a hypothesis on `tst2018` were computed via the evaluation server of the original task.

B.5 IWSLT 2014 Slovenian-English MT Track

Table B.5: IWSLT 2014 Slovenian↔English MT track corpora statistics.

Name	Domain		Slovenian	English
`train-bilingual`	TED talk	Sentence pairs	17k	
		Word vocabulary	36k	18k
		Running words	283k	337k
		Subword vocabulary	23k	16k
		Running subwords	316k	346k
`train-monolingual`	wikipedia	Sentences	1.9M	-
		Running words	36M	-
		Running subwords	42M	-
`dev2012`	TED talk	Sentence pairs	1144	
		Running words	19k	23k
		OOV words	1852 (9.7%)	937 (4.1%)
		Running subwords	21k	23k
		OOV subwords	733 (3.4%)	667 (2.8%)
`tst2012`	TED talk	Sentence pairs	1411	
		Running words	21k	25k
		OOV words	1821 (8.7%)	936 (3.7%)
		Running subwords	24k	26k
		OOV subwords	746 (3.2%)	684 (2.6%)

- Original task: `https://sites.google.com/site/iwsltevaluation2014/mt-track`
- Subword units: separate BPE with 20k (source) or 50k (target) merge operations, no vocabulary threshold

B.6 Belarusian-English TED Talk Translation Task

Table B.6: Belarusian↔English TED talk translation task corpora statistics.

Name	Domain		Belarusian	English
`train-bilingual`	TED talk	Sentence pairs	4.5k	
		Word vocabulary	15k	7.8k
		Running words	75k	91k
		Subword vocabulary	9.1k	8.2k
		Running subwords	100k	96k
`train-monolingual`	wikipedia	Sentences	1.7M	-
		Running words	29M	-
		Running subwords	38M	-
`dev`	TED talk	Sentence pairs	248	
		Running words	4.1k	4.5k
		OOV words	540 (13%)	248 (5.5%)
		Running subwords	5.3k	4.7k
		OOV subwords	201 (3.8%)	229 (4.9%)
`test`	TED talk	Sentence pairs	664	
		Running words	12k	14k
		OOV words	1844 (16%)	886 (6.2%)
		Running subwords	15k	15k
		OOV subwords	790 (5.1%)	821 (5.5%)

- Original task: `https://github.com/neulab/word-embeddings-for-nmt` [Qi & Sachan^{+} 18]
- Subword units: separate BPE with 20k (source) or 50k (target) merge operations, no vocabulary threshold

B.7 Azerbaijani-English TED Talk Translation Task

Table B.7: Azerbaijani↔English TED talk translation task corpora statistics.

Name	Domain		Azerbaijani	English
`train-bilingual`	TED talk	Sentence pairs	5.9k	
		Word vocabulary	18k	8.6k
		Running words	90k	125k
		Subword vocabulary	10k	8.9k
		Running subwords	121k	131k
`train-monolingual`	wikipedia	Sentences	1.8M	-
		Running words	29M	-
		Running subwords	39M	-
`dev`	TED talk	Sentence pairs	671	
		Running words	9.4k	13k
		OOV words	1634 (17%)	672 (5.1%)
		Running subwords	13k	14k
		OOV subwords	422 (3.3%)	587 (4.2%)
`test`	TED talk	Sentence pairs	903	
		Running words	13k	19k
		OOV words	2350 (18%)	1116 (5.9%)
		Running subwords	18k	20k
		OOV subwords	648 (3.6%)	1011 (5.1%)

- Original task: `https://github.com/neulab/word-embeddings-for-nmt` [Qi & Sachan^{+} 18]
- Subword units: separate BPE with 20k (source) or 50k (target) merge operations, no vocabulary threshold

B.8 WMT 2019 German-Czech News Translation Task

This task is originally limited to unsupervised learning (using only monolingual corpora) in WMT 2019, but we relaxed this constraint by the available parallel data (News Commentary v14, `newstest2008-2010`).

Table B.8: WMT 2019 German↔Czech news translation task corpora statistics.

Name	Domain		German	Czech
`train-bilingual`	news commentary	Sentence pairs	226k	
		Word vocabulary	169k	179k
		Running words	5.7M	5.1M
		Subword vocabulary	32181	
		Running subwords	6.8M	6.3M
`newstest2011`	news article	Sentence pairs	3003	
		Running words	73k	65k
		OOV words	4623 (6.4%)	4248 (6.5%)
		Running subwords	99k	92k
		OOV subwords	25 (0.0%)	1 (0.0%)
`newstest2012`	news article	Sentence pairs	3003	
		Running words	73k	65k
		OOV words	4230 (5.8%)	3789 (5.8%)
		Running subwords	97k	90k
		OOV subwords	11 (0.0%)	2 (0.0%)
`newstest2013`	news article	Sentence pairs	3000	
		Running words	63k	57k
		OOV words	3124 (4.9%)	3068 (5.4%)
		Running subwords	83k	78k
		OOV subwords	14 (0.0%)	2 (0.0%)

- Original task: `http://www.statmt.org/wmt19/translation-task.html`
- Subword units: joint BPE with 32k merge operations, no vocabulary threshold

B.9 WMT 2019 French-German News Translation Task

Table B.9: WMT 2019 French↔German news translation task corpora statistics.

Name	Domain		French	German
`train-bilingual`	parliament speech, website crawl, news commentary, news article	Sentence pairs	2.5M	
		Word vocabulary	376k	835k
		Running words	72M	62M
		Subword vocabulary		
		Running subwords	79M	73M
`newstest2011`	news article	Sentence pairs	3003	
		Running words	85k	73k
		OOV words	1306 (1.5%)	2299 (3.2%)
		Running subwords	100k	97k
		OOV subwords	0 (0.0%)	0 (0.0%)
`newstest2012`	news article	Sentence pairs	3003	
		Running words	82k	73k
		OOV words	1009 (1.2%)	2064 (2.8%)
		Running subwords	96k	95k
		OOV subwords	0 (0.0%)	1 (0.0%)
`newstest2013`	news article	Sentence pairs	3000	
		Running words	74k	63k
		OOV words	864 (1.2%)	1507 (2.4%)
		Running subwords	86k	81k
		OOV subwords	0 (0.0%)	0 (0.0%)

- Original task: `http://www.statmt.org/wmt19/translation-task.html`
- Subword units: joint BPE with 32k merge operations, no vocabulary threshold

B.10 WMT 2019 English-Czech News Translation Task

Table B.10: WMT 2019 English↔Czech news translation task corpora statistics.

Name	Domain		English	Czech
`train-bilingual`	parliament speech, website crawl, news, literature, movie, wikipedia title, legislation	Sentence pairs	49M	
		Word vocabulary	1.6M	2.7M
		Running words	690M	613M
		Subword vocabulary	33925	33986
		Running subwords	731M	718M
`newstest2015`	news article	Sentence pairs	2656	
		Running words	54k	46k
		OOV words	350 (0.6%)	456 (1.0%)
		Running subwords	60k	59k
		OOV subwords	0 (0.0%)	0 (0.0%)
`newstest2016`	news article	Sentence pairs	2999	
		Running words	65k	57k
		OOV words	612 (0.9%)	737 (1.3%)
		Running subwords	73k	73k
		OOV subwords	0 (0.0%)	0 (0.0%)
`newstest2017`	news article	Sentence pairs	3005	
		Running words	62k	55k
		OOV words	447 (0.7%)	640 (1.2%)
		Running subwords	69k	70k
		OOV subwords	0 (0.0%)	0 (0.0%)

- Original task: `http://www.statmt.org/wmt19/translation-task.html`
- Subword units: separate BPE with 32k merge operations, no vocabulary threshold

B.11 WMT 2014 French-English News Translation Task

Table B.11: WMT 2014 French↔English news translation task corpora statistics.

Name	Domain		French	English
`train-bilingual`	parliamentary content, website crawl, news	Sentence pairs	35M	
		Word vocabulary	2.8M	2.9M
		Running words	1.1B	960M
		Subword vocabulary	35227	35352
		Running subwords	1.2B	1.0B
`train-monolingual`	news article	Sentences	42M	100M
		Running words	1.0B	2.3B
		Running subwords		
`newstest2012`	news article	Sentence pairs	3003	
		Running words	82k	73k
		OOV words	490 (0.6%)	447 (0.6%)
		Running subwords	92k	81k
		OOV subwords	0 (0.0%)	0 (0.0%)
`newstest2013`	news article	Sentence pairs	3000	
		Running words	74k	65k
		OOV words	411 (0.6%)	361 (0.6%)
		Running subwords	82k	72k
		OOV subwords	0 (0.0%)	0 (0.0%)
`newstest2014`	news article	Sentence pairs	3003	
		Running words	81k	71k
		OOV words	288 (0.4%)	326 (0.5%)
		Running subwords	91k	80k
		OOV subwords	0 (0.0%)	0 (0.0%)

- Original task: `http://www.statmt.org/wmt15/translation-task.html`
- Subword units: separate BPE with 32k merge operations, no vocabulary threshold

B.12 EuTrans Spanish-English Decipherment Task

This task was prepared specifically for decipherment without a reordering problem.

Table B.12: EuTrans Spanish-English Decipherment Task corpora statistics.

Name	Domain		Spanish	English
`monolingual`	travel	Sentences	10k	488k
		Word vocabulary	677	505
		Running words	85k	4.2M

- Original task: [Amengual & Benedí+ 96]

B.13 Europarl Spanish-English Decipherment Task

This task was prepared in the same way as Section B.12. In addition, we left out long sentences with more than 25 words and sentences with singletons to further reduce the complexity.

Table B.13: Europarl Spanish-English Decipherment Task corpora statistics.

Name	Domain		Spanish	English
`monolingual`	parliament speech	Sentences	182k	1.7M
		Word vocabulary	32k	96k
		Running words	2.7M	43M

- Original data: `https://www.statmt.org/europarl/`

B.14 WMT 2018 Russian-English News Translation Task

Table B.14: WMT 2018 Russian↔English news translation task corpora statistics.

Name	Domain		Russian	English
`train-bilingual`	news article, news commentary, website crawl	Sentence pairs	25M	
		Word vocabulary	2.6M	2.7M
		Running words	414M	410M
		Subword vocabulary	33023	
		Running subwords	881M	790M
`train-monolingual`	news article	Sentences	72M	72M
		Running words	1.4B	1.6B
		Running subwords	1.7B	1.9B
`newstest2016`	news article	Sentence pairs	2998	
		Running words	62k	70k
		OOV words	702 (1.1%)	460 (0.7%)
		Running subwords	84k	81k
		OOV subwords	1 (0.0%)	7 (0.0%)
`newstest2017`	news article	Sentence pairs	3001	
		Running words	60k	70k
		OOV words	701 (1.2%)	564 (0.8%)
		Running subwords	83k	82k
		OOV subwords	0 (0.0%)	0 (0.0%)
`newstest2018`	news article	Sentence pairs	3000	
		Running words	62k	73k
		OOV words	765 (1.2%)	603 (0.8%)
		Running subwords	86k	85k
		OOV subwords	0 (0.0%)	2 (0.0%)

- Original task: `http://www.statmt.org/wmt18/translation-task.html`
- Subword units: joint BPE with 32k merge operations, no vocabulary threshold

B.15 WMT 2019 Chinese-English News Translation Task

Table B.15: WMT 2019 Chinese↔English news translation task corpora statistics.

Name	Domain		Chinese	English
`train-bilingual`	parliamentary content, wikipedia title, news, website crawl, textbook, law, technical	Sentence pairs	19M	
		Word vocabulary	2.1M	1.5M
		Running words	440M	483M
		Subword vocabulary	48746	
		Running subwords	500M	557M
`train-monolingual`	news article	Sentences	31M	31M
		Running words	1.4B	699M
		Running subwords	1.6B	813M
`newsdev2017`	news article	Sentence pairs	2002	
		Running words	52k	59k
		OOV words	550 (1.1%)	340 (0.6%)
		Running subwords	61k	67k
		OOV subwords	10 (0.0%)	1 (0.0%)
`newstest2018`	news article	Sentence pairs	3981	
		Running words	93k	112k
		OOV words	1179 (1.3%)	725 (0.6%)
		Running subwords	110k	127k
		OOV subwords	8 (0.0%)	2 (0.0%)

- Original task: `http://www.statmt.org/wmt19/translation-task.html`
- Subword units: joint BPE with 32k merge operations, no vocabulary threshold

B.16 WMT 2019 Kazakh-English News Translation Task

Table B.16: WMT 2019 Kazakh↔English news translation task corpora statistics.

Name	Domain		Kazakh	English
`train-bilingual`	news commentary, wikipedia title, television script	Sentence pairs	222k	
		Word vocabulary	196k	130k
		Running words	1.6M	1.9M
		Subword vocabulary	30868	
		Running subwords	2.7M	2.8M
`train-monolingual`	news article	Sentences	18.5M	18.5M
		Running words	279M	421M
		Running subwords	413M	487M
`newsdev2019`	news article	Sentence pairs	2066	
		Running words	43k	53k
		OOV words	2169 (5.1%)	1113 (2.1%)
		Running subwords	62k	61k
		OOV subwords	40 (0.1%)	55 (0.1%)
`newstest2019`	news article	Sentence pairs	1000	
		Running words	16k	20k
		OOV words	1437 (9.2%)	717 (3.5%)
		Running subwords	23k	25k
		OOV subwords	45 (0.2%)	33 (0.1%)

- Original task: `http://www.statmt.org/wmt19/translation-task.html`
- Subword units: joint BPE with 32k merge operations, no vocabulary threshold

B.17 WMT 2019 Gujarati-English News Translation Task

Table B.17: WMT 2019 Gujarati↔English news translation task corpora statistics.

Name	Domain		Gujarati	English
`train-bilingual`	news commentary, wikipedia, bible, software, website crawl	Sentence pairs	156k	
		Word vocabulary	85k	58k
		Running words	2.3M	1.5M
		Subword vocabulary	29076	
		Running subwords	2.7M	1.9M
`train-monolingual`	news article	Sentences	4.1M	4.1M
		Running words	121M	94M
		Running subwords	151M	107M
`newsdev2019`	news article	Sentence pairs	1998	
		Running words	58k	42k
		OOV words	3953 (6.8%)	2808 (6.6%)
		Running subwords	71k	49k
		OOV subwords	131 (0.2%)	608 (1.2%)
`newstest2019`	news article	Sentence pairs	1016	
		Running words	22k	18k
		OOV words	1864 (8.3%)	818 (4.6%)
		Running subwords	27k	21k
		OOV subwords	80 (0.3%)	123 (0.6%)

- Original task: `http://www.statmt.org/wmt19/translation-task.html`
- Subword units: joint BPE with 32k merge operations, no vocabulary threshold

List of Figures

List of Tables

Bibliography

[Abadi & Barham$^+$ 16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng: Tensorflow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016)*, pp. 265–283, Savannah, GA, USA, November 2016.

[Aharoni & Johnson$^+$ 19] R. Aharoni, M. Johnson, O. Firat: Massively Multilingual Neural Machine Translation. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pp. 3874–3884, Minneapolis, MN, USA, June 2019.

[Alkhouli 20] T. Alkhouli: *Alignment-Based Neural Networks for Machine Translation.* Ph.D. thesis, Department of Computer Science, RWTH Aachen University, Aachen, Germany, 2020.

[Alkhouli & Guta$^+$ 14] T. Alkhouli, A. Guta, H. Ney: Vector Space Models for Phrase-based Machine Translation. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, pp. 1–10, Doha, Qatar, October 2014.

[Amengual & Benedí$^+$ 96] J.C. Amengual, J.M. Benedí, A. Castaño, A. Marzal, F. Prat, E. Vidal, J.M. Vilar, C. Delogu, A.D. Carlo, H. Ney, S. Vogel: Definition of a Machine Translation Task and Generation of Corpora. Technical report, EuTrans (IT-LTR-OS-20268), 1996.

[Arivazhagan & Bapna$^+$ 19] N. Arivazhagan, A. Bapna, O. Firat, R. Aharoni, M. Johnson, W. Macherey: The Missing Ingredient in Zero-Shot Neural Machine Translation. arXiv cs.CL 1903.07091, 2019.

[Artetxe & Labaka$^+$ 16] M. Artetxe, G. Labaka, E. Agirre: Learning Principled Bilingual Mappings of Word Embeddings while Preserving Monolingual Invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pp. 2289–2294, Austin, TX, USA, November 2016.

[Artetxe & Labaka$^+$ 17] M. Artetxe, G. Labaka, E. Agirre: Learning Bilingual Word Embeddings with (Almost) No Bilingual Data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pp. 451–462, Vancouver, Canada, July 2017.

[Artetxe & Labaka$^+$ 18] M. Artetxe, G. Labaka, E. Agirre, K. Cho: Unsupervised Neural Machine Translation. In *Proceedings of Sixth International Conference on Learning Representations (ICLR 2018)*, Vancouver, Canada, May 2018.

[Artetxe & Labaka$^+$ 19] M. Artetxe, G. Labaka, E. Agirre: An Effective Approach to Unsupervised Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pp. 194–203, Florence, Italy, July 2019.

[Auli & Galley$^+$ 13] M. Auli, M. Galley, C. Quirk, G. Zweig: Joint Language and Translation Modeling with Recurrent Neural Networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pp. 1044–1054, Seattle, WA, USA, October 2013.

[Ba & Kiros$^+$ 16] J.L. Ba, J.R. Kiros, G.E. Hinton: Layer Normalization. arXiv stat.ML 1607.06450, 2016.

[Bahdanau & Cho$^+$ 15] D. Bahdanau, K. Cho, Y. Bengio: Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the Third International Conference on Learning Representations (ICLR 2015)*, San Diego, CA, USA, May 2015.

[Bahl & Jelinek$^+$ 83] L.R. Bahl, F. Jelinek, R.L. Mercer: A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 5, No. 2, pp. 179–190, 1983.

[Bakhshaei & Khadivi$^+$ 10] S. Bakhshaei, S. Khadivi, N. Riahi: Farsi-German Statistical Machine Translation through Bridge Language. In *Proceedings of the Fifth International Symposium on Telecommunications (IST 2010)*, pp. 557–561, Kish Island, Iran, December 2010.

[Barrault & Bojar$^+$ 19] L. Barrault, O. Bojar, M.R. Costa-jussà, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, P. Koehn, S. Malmasi, C. Monz, M. Müller, S. Pal, M. Post, M. Zampieri: Findings of the 2019 Conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (WMT 2019)*, pp. 1–61, Florence, Italy, August 2019.

[Baziotis & Haddow$^+$ 20] C. Baziotis, B. Haddow, A. Birch: Language Model Prior for Low-Resource Neural Machine Translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pp. 7622–7634, Virtual Conference, November 2020.

[Bengio 12] Y. Bengio: Practical Recommendations for Gradient-based Training of Deep Architectures. In *Neural Networks: Tricks of the Trade*, pp. 437–478. Springer, 2012.

[Bengio & Ducharme$^+$ 03] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin: A Neural Probabilistic Language Model. *Journal of Machine Learning Research (JMLR)*, Vol. 3, No. Feb, pp. 1137–1155, 2003.

[Bentivogli & Bisazza$^+$ 16] L. Bentivogli, A. Bisazza, M. Cettolo, M. Federico: Neural versus Phrase-Based Machine Translation Quality: a Case Study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pp. 257–267, Austin, TX, USA, November 2016.

[Berg-Kirkpatrick & Klein 13] T. Berg-Kirkpatrick, D. Klein: Decipherment with a Million Random Restarts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pp. 874–878, Seattle, WA, USA, October 2013.

[Berger & Pietra$^+$ 96] A.L. Berger, S.A.D. Pietra, V.J.D. Pietra: A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, Vol. 22, No. 1, pp. 39–71, March 1996.

[Bertoldi & Barbaiani$^+$ 08] N. Bertoldi, M. Barbaiani, M. Federico, R. Cattoni: Phrase-based Statistical Machine Translation with Pivot Languages. In *Proceedings of Fifth International Workshop on Spoken Language Translation (IWSLT 2008)*, pp. 143–149, Honolulu, HI, USA, October 2008.

[Bishop 06] C.M. Bishop: *Pattern Recognition and Machine Learning.* Springer, New York, NY, USA, 2006.

[Bojanowski & Grave$^+$ 17] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov: Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics (TACL)*, Vol. 5, pp. 135–146, 2017.

[Bojar & Chatterjee$^+$ 17] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, S. Huang, M. Huck, P. Koehn, Q. Liu, V. Logacheva, C. Monz, M. Negri, M. Post, R. Rubino, L. Specia, M. Turchi: Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation (WMT 2017)*, pp. 169–214, Copenhagen, Denmark, September 2017.

[Bojar & Federmann$^+$ 18] O. Bojar, C. Federmann, M. Fishel, Y. Graham, B. Haddow, P. Koehn, C. Monz: Findings of the 2018 Conference on Machine Translation (WMT18). In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 272–303, Brussels, Belgium, October 2018.

[Borthwick & Sterling$^+$ 98] A. Borthwick, J. Sterling, E. Agichtein, R. Grishman: Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition. In *Proceedings of the Sixth Workshop on Very Large Corpora (VLC 1998)*, pp. 152–160, Montreal, Canada, August 1998.

[Bourlard & Morgan 94] H. Bourlard, N. Morgan: *Connectionist Speech Recognition: A Hybrid Approach.* Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[Brown & Cocke$^+$ 90] P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, P.S. Roossin: A Statistical Approach to Machine Translation. *Computational Linguistics*, Vol. 16, No. 2, pp. 79–85, 1990.

[Brown & Della Pietra$^+$ 93] P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, R.L. Mercer: The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, Vol. 19, No. 2, pp. 263–311, 1993.

[Brown & deSouza$^+$ 92] P.F. Brown, P.V. deSouza, R.L. Mercer, V.J.D. Pietra, J.C. Lai: Class-Based n-gram Models of Natural Language. *Computational Linguistics*, Vol. 18, No. 4, pp. 467–479, December 1992.

[Burlot & Yvon 18] F. Burlot, F. Yvon: Using Monolingual Data in Neural Machine Translation: a Systematic Study. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 144–155, Brussels, Belgium, November 2018.

[Cao & Zhao$^+$ 16] H. Cao, T. Zhao, S. Zhang, Y. Meng: A Distribution-based Model to Learn Bilingual Word Embeddings. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*, pp. 1818–1827, Osaka, Japan, December 2016.

[Caruana 97] R. Caruana: Multitask Learning. *Machine Learning*, Vol. 28, No. 1, pp. 41–75, 1997.

[Cer & Manning$^+$ 10] D. Cer, C.D. Manning, D. Jurafsky: The Best Lexical Metric for Phrase-Based Statistical MT System Optimization. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2010)*, pp. 555–563, Los Angeles, CA, USA, June 2010.

[Chen & Goodman 99] S.F. Chen, J. Goodman: An Empirical Study of Smoothing Techniques for Language Modeling. *Computer Speech & Language*, Vol. 13, No. 4, pp. 359–394, 1999.

[Chen & Li$^+$ 15] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, Z. Zhang: MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. arXiv cs.DC 1512.01274, 2015.

[Chen & Liu$^+$ 17] Y. Chen, Y. Liu, Y. Cheng, V.O. Li: A Teacher-Student Framework for Zero-Resource Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pp. 1925–1935, Vancouver, Canada, July 2017.

[Chen & Zhang$^+$ 08] B. Chen, M. Zhang, A. Aw, H. Li: Exploiting N-best Hypotheses for SMT Self-Enhancement. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL 2008)*, pp. 157–160, Columbus, OH, USA, June 2008.

[Cheng & Yang$^+$ 17] Y. Cheng, Q. Yang, Y. Liu, M. Sun, W. Xu: Joint Training for Pivot-based Neural Machine Translation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pp. 3974–3980, Melbourne, Australia, August 2017.

[Chiang & Knight$^+$ 09] D. Chiang, K. Knight, W. Wang: 11,001 New Features for Statistical Machine Translation. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2009)*, pp. 218–226, Boulder, CO, USA, May 2009.

[Church & Gale 91] K.W. Church, W.A. Gale: Probability Scoring for Spelling Correction. *Statistics and Computing*, Vol. 1, No. 2, pp. 93–103, 1991.

[Cichon & Gan 15] J. Cichon, W.B. Gan: Branch-specific Dendritic Ca 2+ Spikes Cause Persistent Synaptic Plasticity. *Nature*, Vol. 520, No. 7546, pp. 180–185, 2015.

[Cisse & Bojanowski$^+$ 17] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, N. Usunier: Parseval Networks: Improving Robustness to Adversarial Examples. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, pp. 854–863, Sydney, Austrailia, August 2017.

[Clark & Dyer$^+$ 11] J.H. Clark, C. Dyer, A. Lavie, N.A. Smith: Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011)*, pp. 176–181, Portland, OR, USA, June 2011.

[Collobert & Weston 08] R. Collobert, J. Weston: A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pp. 160–167, Helsinki, Finland, July 2008.

[Collobert & Weston$^+$ 11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa: Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research (JMLR)*, Vol. 12, pp. 2493–2537, 2011.

[Conneau & Lample$^+$ 18] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, H. Jégou: Word Translation without Parallel Data. In *Proceedings of Sixth International Conference on Learning Representations (ICLR 2018)*, Vancouver, Canada, May 2018.

[Conneau & Lample 19] A. Conneau, G. Lample: Cross-lingual Language Model Pretraining. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pp. 7057–7067, Vancouver, Canada, December 2019.

[Costa-Jussa & Farrús$^+$ 12] M.R. Costa-Jussa, M. Farrús, J.B. Mariño, J.A. Fonollosa: Study and Comparison of Rule-Based and Statistical Catalan-Spanish Machine Translation Systems. *Computing and Informatics*, Vol. 31, No. 2, pp. 245–270, 2012.

[Costa-Jussà & Henríquez$^+$ 11] M.R. Costa-Jussà, C. Henríquez, R.E. Banchs: Enhancing scarce-resource language translation through pivot combinations. In *Proceedings of Fifth International Joint Conference on Natural Language Processing (IJCNLP 2011)*, pp. 1361–1365, Chiang Mai, Thailand, November 2011.

[Cotterell & Kreutzer 18] R. Cotterell, J. Kreutzer: Explaining and Generalizing Back-Translation through Wake-Sleep. arXiv cs.CL 1806.04402, 2018.

[Crammer & Kulesza$^+$ 09] K. Crammer, A. Kulesza, M. Dredze: Adaptive Regularization of Weight Vectors. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pp. 414–422, Vancouver, Canada, December 2009.

[Crammer & Singer 03] K. Crammer, Y. Singer: Ultraconservative Online Algorithms for Multi-class Problems. *Journal of Machine Learning Research (JMLR)*, Vol. 3, No. Jan, pp. 951–991, 2003.

[Currey & Barone$^+$ 17] A. Currey, A.V.M. Barone, K. Heafield: Copied Monolingual Data Improves Low-Resource Neural Machine Translation. In *Proceedings of the Second Conference on Machine Translation (WMT 2017)*, pp. 148–156, Copenhagen, Denmark, September 2017.

[Dairoch & Ratcliff 72] J.N. Dairoch, D. Ratcliff: Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, Vol. 43, No. 5, pp. 1470–1480, 1972.

[De Gispert & Marino 06] A. De Gispert, J.B. Marino: Catalan-English statistical machine translation without parallel corpus: bridging through Spanish. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pp. 65–68, Genoa, Italy, May 2006.

[Deligne & Bimbot 95] S. Deligne, F. Bimbot: Improved Backing-off for m-gram Language Modeling. In *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1995)*, pp. 181–184, Detroit, MI, USA, May 1995.

[Dempster & Laird$^+$ 77] A.P. Dempster, N.M. Laird, D.B. Rubin: Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, Vol. 39, No. 1, pp. 1–38, 1977.

[Deng & Cheng$^+$ 18] Y. Deng, S. Cheng, J. Lu, K. Song, J. Wang, S. Wu, L. Yao, G. Zhang, H. Zhang, P. Zhang et al.: Alibaba's neural machine translation systems for wmt18. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 368–376, Brussels, Belgium, November 2018.

[Devlin & Chang$^+$ 19] J. Devlin, M.W. Chang, K. Lee, K. Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pp. 4171–4186, Minneapolis, MN, USA, June 2019.

[Devlin & Zbib$^+$ 14] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, J. Makhoul: Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pp. 1370–1380, Baltimore, MD, USA, June 2014.

[Dietterich 00] T.G. Dietterich: Ensemble Methods in Machine Learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems (MCS 2000)*, pp. 1–15, Cagliari, Italy, June 2000.

[Doetsch & Kozielski$^+$ 14] P. Doetsch, M. Kozielski, H. Ney: Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition. In *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR 2014)*, pp. 279–284, Crete, Greece, September 2014.

[Doetsch & Zeyer$^+$ 17] P. Doetsch, A. Zeyer, P. Voigtlaender, I. Kulikov, R. Schlüter, H. Ney: RETURNN: The RWTH Extensible Training Framework for Universal Recurrent Neural Networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*, pp. 5345–5349, New Orleans, LA, USA, March 2017.

[Dong & Wu$^+$ 15] D. Dong, H. Wu, W. He, D. Yu, H. Wang: Multi-Task Learning for Multiple Language Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the Seventh International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, pp. 1723–1732, Beijing, China, July 2015.

[Dou & Knight 12] Q. Dou, K. Knight: Large Scale Decipherment for Out-of-Domain Machine Translation. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Computational Language Learning (EMNLP-CoNLL 2012)*, pp. 266–275, Jeju, Republic of Korea, July 2012.

[Dou & Knight 13] Q. Dou, K. Knight: Dependency-based Decipherment for Resource-Limited Machine Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pp. 1668–1676, Seattle, WA, USA, October 2013.

[Dou & Vaswani$^+$ 15] Q. Dou, A. Vaswani, K. Knight: Unifying Bayesian Inference and Vector Space Models for Improved Decipherment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the Seventh International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, pp. 836–845, Beijing, China, July 2015.

[Ștefănescu & Ion$^+$ 12] D. Ștefănescu, R. Ion, S. Hunsicker: Hybrid Parallel Sentence Mining from Comparable Corpora. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT 2012)*, pp. 137–144, Trento, Italy, May 2012.

[Duchi & Hazan$^+$ 11] J. Duchi, E. Hazan, Y. Singer: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research (JMLR)*, Vol. 12, No. Jul, pp. 2121–2159, 2011.

[Dyer & Chahuneau$^+$ 13] C. Dyer, V. Chahuneau, N.A. Smith: A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pp. 644–648, Atlanta, GA, USA, June 2013.

[Echihabi & Marcu 03] A. Echihabi, D. Marcu: A Noisy-Channel Approach to Question Answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pp. 16–23, Sapporo, Japan, July 2003.

[Edunov & Ott$^+$ 18] S. Edunov, M. Ott, M. Auli, D. Grangier: Understanding Back-Translation at Scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pp. 489–500, Brussels, Belgium, October 2018.

[Espanã-Boquera & Castro-Bleda$^+$ 11] S. Espanã-Boquera, M.J. Castro-Bleda, J. Gorbe-Moya, F. Zamora-Martinez: Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 33, No. 4, pp. 767–779, April 2011.

[Esplá-Gomis & Forcada 09] M. Esplá-Gomis, M.L. Forcada: Bitextor, a Free/Open-Source Software to Harvest Translation Memories from Multilingual Websites. In *Proceedings of the 12th Machine Translation Summit (MT Summit XII)*, Ottawa, Canada, August 2009.

[Esplà-Gomis & Forcada$^+$ 19] M. Esplà-Gomis, M.L. Forcada, G. Ramírez-Sánchez, H. Hoang: ParaCrawl: Web-Scale Parallel Corpora for the Languages of the EU. In *Proceedings of the 17th Machine Translation Summit (MT Summit XVII)*, pp. 118–119, Dublin, Ireland, August 2019.

[Etchegoyhen & Azpeitia 16] T. Etchegoyhen, A. Azpeitia: Set-theoretic Alignment for Comparable Corpora. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pp. 2009–2018, Berlin, Germany, July 2016.

[Fadaee & Monz 18] M. Fadaee, C. Monz: Back-Translation Sampling by Targeting Difficult Words in Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pp. 436–446, Brussels, Belgium, November 2018.

[Firat & Cho$^+$ 16] O. Firat, K. Cho, Y. Bengio: Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism. In *Proceedings of the 15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pp. 866–875, San Diego, CA, USA, January 2016.

[Firat & Sankaran$^+$ 16] O. Firat, B. Sankaran, Y. Al-Onaizan, F.T.Y. Vural, K. Cho: Zero-Resource Translation with Multi-Lingual Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pp. 268–277, Austin, TX, USA, November 2016.

[Foster & Kuhn$^+$ 06] G. Foster, R. Kuhn, H. Johnson: Phrasetable Smoothing for Statistical Machine Translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pp. 53–61, Sydney, Austrailia, July 2006.

[Freitag 10] M. Freitag: Minimum Error Rate Training Extensions for Statistical Machine Translation. Master's thesis, Department of Computer Science, RWTH Aachen University, Aachen, Germany, March 2010.

[Freitag & Al-Onaizan 16] M. Freitag, Y. Al-Onaizan: Fast Domain Adaptation for Neural Machine Translation. arXiv cs.CL 1612.06897, 2016.

[Gehring & Auli$^+$ 17] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y.N. Dauphin: Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, pp. 1243–1252, Sydney, Austrailia, August 2017.

[Geng 18] J. Geng: Unsupervised Learning of Neural Network Lexicon and Cross-lingual Word Embedding. Master's thesis, Department of Computer Science, RWTH Aachen University, Aachen, Germany, 2018.

[Goldwater & Griffiths 07] S. Goldwater, T.L. Griffiths: A Fully Bayesian Approach to Unsupervised Part-of-Speech Tagging. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, pp. 744–751, Prague, Czech Republic, June 2007.

[Goodfellow & Bengio$^+$ 16] I. Goodfellow, Y. Bengio, A. Courville: *Deep Learning.* MIT Press, November 2016.

[Goodfellow & Mirza$^+$ 14] I.J. Goodfellow, M. Mirza, A.C. Da Xiao, Y. Bengio: An Empirical Investigation of Catastrophic Forgetting in Gradient-based Neural Networks. In *Proceedings of the First International Conference on Learning Representations (ICLR 2014)*, Banff, Canada, April 2014.

[Goodfellow & Pouget-Abadie$^+$ 14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio: Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pp. 2672–2680, Montréal, Canada, December 2014.

[Graça & Kim$^+$ 18] M. Graça, Y. Kim, J. Schamper, J. Geng, H. Ney: The RWTH Aachen University English-German and German-English Unsupervised Neural Machine Translation Systems for WMT 2018. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 377–385, Belgium, Brussels, October 2018.

[Graça & Kim$^+$ 19] M. Graça, Y. Kim, J. Schamper, S. Khadivi, H. Ney: Generalizing Back-Translation in Neural Machine Translation. In *Proceedings of the Fourth Conference on Machine Translation (WMT 2019)*, pp. 45–52, Florence, Italy, August 2019.

[Graça 19] M. Graça: Investigations in Semi-supervised Neural Machine Translation. Master's thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, January 2019.

[Graham & Haddow$^+$ 19] Y. Graham, B. Haddow, P. Koehn: Translationese in Machine Translation Evaluation. arXiv cs.CL 1906.09833, 2019.

[Graves 12] A. Graves: *Supervised Sequence Labelling with Recurrent Neural Networks.* Springer, 2012.

[Graves & Mohamed$^+$ 13] A. Graves, A. Mohamed, G. Hinton: Speech Recognition with Deep Recurrent Neural Networks. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013)*, pp. 6645–6649, Vancouver, Canada, May 2013.

[Green & Wang$^+$ 13] S. Green, S. Wang, D. Cer, C.D. Manning: Fast and Adaptive Online Training of Feature-Rich Translation Models. In *Proceedings of the 51st Annual Meeting of the*

Association for Computational Linguistics (ACL 2013), pp. 311–321, Sofia, Bulgaria, August 2013.

[Gulcehre & Firat$^+$ 15] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.C. Lin, F. Bougares, H. Schwenk, Y. Bengio: On Using Monolingual Corpora in Neural Machine Translation. arXiv cs.CL 1503.03535, 2015.

[Guta & Alkhouli$^+$ 15] A. Guta, T. Alkhouli, J.T. Peter, J. Wuebker, H. Ney: A Comparison Between Count and Neural Network Models Based on Joint Translation and Reordering Sequences. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pp. 1401–1411, Lisbon, Portugal, September 2015.

[Guzmán & Chen$^+$ 19] F. Guzmán, P.J. Chen, M. Ott, J. Pino, G. Lample, P. Koehn, V. Chaudhary, M. Ranzato: The FLORES Evaluation Datasets for Low-Resource Machine Translation: Nepali–English and Sinhala–English. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the Ninth International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*, pp. 6100–6113, Hong Kong, China, November 2019.

[Ha & Niehues$^+$ 17] T.L. Ha, J. Niehues, A. Waibel: Effective Strategies in Zero-Shot Neural Machine Translation. In *Proceedings of the 14th International Workshop on Spoken Language Translation (IWSLT 2017)*, Tokyo, Japan, December 2017.

[Hasan & Ganitkevitch$^+$ 08] S. Hasan, J. Ganitkevitch, H. Ney, J. Andrés-Ferrer: Triplet Lexicon Models for Statistical Machine Translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pp. 372–381, Honolulu, HI, USA, October 2008.

[Hassan & Aue$^+$ 18] H. Hassan, A. Aue, C. Chen, V. Chowdhary, J. Clark, C. Federmann, X. Huang, M. Junczys-Dowmunt, W. Lewis, M. Li, S. Liu, T.Y. Liu, R. Luo, A. Menezes, T. Qin, F. Seide, X. Tan, F. Tian, L. Wu, S. Wu, Y. Xia, D. Zhang, Z. Zhang, M. Zhou: Achieving Human Parity on Automatic Chinese to English News Translation. arXiv cs.CL 1803.05567, 2018.

[Hauer & Nicolai$^+$ 17] B. Hauer, G. Nicolai, G. Kondrak: Bootstrapping Unsupervised Bilingual Lexicon Induction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pp. 619–624, Valencia, Spain, April 2017.

[He & Xia$^+$ 16] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T.Y. Liu, W.Y. Ma: Dual Learning for Machine Translation. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pp. 820–828, Barcelona, Spain, December 2016.

[He & Zhang$^+$ 16] K. He, X. Zhang, S. Ren, J. Sun: Deep Residual Learning for Image Recognition. In *Proceedings of the 2016 Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[Heafield 11] K. Heafield: KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT 2011)*, pp. 187–197, Edinburgh, Scotland, July 2011.

[Hieber & Domhan$^+$ 17] F. Hieber, T. Domhan, M. Denkowski, D. Vilar, A. Sokolov, A. Clifton, M. Post: Sockeye: A Toolkit for Neural Machine Translation. arXiv cs.CL 1712.05690, 2017.

[Hill & Cho$^+$ 16] F. Hill, K. Cho, A. Korhonen: Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pp. 1367–1377, San Diego, CA, USA, January 2016.

[Hilmes 20] B. Hilmes: Investigation on the Model Architecture for Pivot-based Neural Machine Translation. Bachelor's thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, July 2020.

[Hirschmann & Nam$^+$ 16] F. Hirschmann, J. Nam, J. Fürnkranz: What Makes Word-level Neural Machine Translation Hard: A Case Study on English-German Translation. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*, pp. 3199–3208, Osaka, Japan, December 2016.

[Hoang & Koehn$^+$ 18] V.C.D. Hoang, P. Koehn, G. Haffari, T. Cohn: Iterative back-translation for neural machine translation. In *Proceedings of the Second Workshop on Neural Machine Translation and Generation (WNGT 2018)*, pp. 18–24, Melbourne, Australia, July 2018.

[Hopkins & May 11] M. Hopkins, J. May: Tuning as Ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pp. 1352–1362, Edinburgh, Scotland, July 2011.

[Hu & Wang$^+$ 07] X. Hu, H. Wang, H. Wu: Using RBMT Systems to Produce Bilingual Corpus for SMT. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pp. 287–295, Prague, Czech Republic, June 2007.

[Huck & Vilar$^+$ 11] M. Huck, D. Vilar, D. Stein, H. Ney: Lightly-supervised Training for Hierarchical Phrase-based Machine Translation. In *Proceedings of the First Workshop on Unsupervised Learning in NLP*, pp. 91–96, Edinburgh, Scotland, July 2011.

[Hutchins 86] W.J. Hutchins: *Machine Translation: Past, Present, Future.* Ellis Horwoord, 1986.

[Imamura & Fujita$^+$ 18] K. Imamura, A. Fujita, E. Sumita: Enhancement of Encoder and Attention Using Target Monolingual Corpora in Neural Machine Translation. In *Proceedings of the Second Workshop on Neural Machine Translation and Generation (WNGT 2018)*, pp. 55–63, Melbourne, Australia, July 2018.

[Johnson 07] M. Johnson: Why Doesn't EM Find Good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pp. 296–305, Prague, Czech Republic, June 2007.

[Johnson & Schuster$^+$ 17] M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado et al.: Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association of Computational Linguistics (TACL)*, Vol. 5, No. 1, pp. 339–351, 2017.

[Kalchbrenner & Blunsom 13] N. Kalchbrenner, P. Blunsom: Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pp. 1700–1709, Seattle, WA, USA, October 2013.

[Kauers & Vogel$^+$ 02] M. Kauers, S. Vogel, C. Fügen, A. Waibel: Interlingua Based Statistical Machine Translation. In *Proceedings of the Seventh International Conference on Spoken Language Processing (ICSLP 2002)*, Denver, CO, USA, September 2002.

[Kaufmann 12] M. Kaufmann: JMaxAlign: A Maximum Entropy Parallel Sentence Alignment Tool. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pp. 277–288, Mumbai, India, December 2012.

[Kernighan & Church$^+$ 90] M.D. Kernighan, K.W. Church, W.A. Gale: A Spelling Correction Program Based on a Noisy Channel Model. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING 1990)*, pp. 205–210, Helsinki, Finland, August 1990.

[Khudanpur & Wu 00] S. Khudanpur, J. Wu: Maximum Entropy Techniques for Exploiting Syntactic, Semantic, and Collocational Dependencies in Language Modeling. *Computer Speech and Language*, Vol. 14, No. 4, pp. 355–372, October 2000.

[Kim 15] Y. Kim: Phrase Table Smoothing with Word Classes. Master's thesis, Department of Computer Science, RWTH Aachen University, Aachen, Germany, July 2015.

[Kim & Gao$^+$ 19] Y. Kim, Y. Gao, H. Ney: Effective Cross-lingual Transfer of Neural Machine Translation Models without Shared Vocabularies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pp. 1246–1257, Florence, Italy, July 2019.

[Kim & Geng$^+$ 18] Y. Kim, J. Geng, H. Ney: Improving Unsupervised Word-by-Word Translation with Language Model and Denoising Autoencoder. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 862–868, Brussels, Belgium, November 2018.

[Kim & Graça$^+$ 20] Y. Kim, M. Graça, H. Ney: When and Why is Unsupervised Neural Machine Translation Useless? In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation (EAMT 2020)*, Lisbon, Portugal, May 2020.

[Kim & Guta$^+$ 16] Y. Kim, A. Guta, J. Wuebker, H. Ney: A Comparative Study on Vocabulary Reduction for Phrase Table Smoothing. In *Proceedings of the First Conference on Machine Translation (WMT 2016)*, pp. 110–117, Berlin, Germany, August 2016.

[Kim & Petrov$^+$ 19] Y. Kim, P. Petrov, P. Petrushkov, S. Khadivi, H. Ney: Pivot-Based Transfer Learning for Neural Machine Translation between Non-English Languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the Ninth International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*, pp. 866–876, Hong Kong, China, November 2019.

[Kim & Rosendahl$^+$ 19] Y. Kim, H. Rosendahl, N. Rossenbach, J. Rosendahl, S. Khadivi, H. Ney: Learning Bilingual Sentence Embeddings via Autoencoding and Computing Similarities with a Multilayer Perceptron. In *Proceedings of the Fourth Workshop on Representation Learning for NLP (RepL4NLP 2019)*, pp. 61–71, Florence, Italy, August 2019.

[Kim & Schamper$^+$ 17] Y. Kim, J. Schamper, H. Ney: Unsupervised Training for Large Vocabulary Translation Using Sparse Lexicon and Word Classes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, 650–656, Valencia, Spain, April 2017.

[Kim & Tran$^+$ 19] Y. Kim, D.T. Tran, H. Ney: When and Why is Document-level Context Useful in Neural Machine Translation? In *Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019)*, pp. 24–34, Hong Kong, China, November 2019.

[Kingma & Ba 15] D.P. Kingma, J. Ba: Adam: A Method for Stochastic Optimization. In *Proceedings of the Third International Conference on Learning Representations (ICLR 2015)*, San Diego, CA, USA, May 2015.

[Kirkpatrick & Pascanu^{+} 17] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, R. Hadsell: Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, Vol. 114, No. 13, pp. 3521–3526, 2017.

[Klein & Kim^{+} 17] G. Klein, Y. Kim, Y. Deng, J. Senellart, A. Rush: OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pp. 67–72, Vancouver, Canada, July 2017.

[Kneser & Ney 95] R. Kneser, H. Ney: Improved Backing-off for m-gram Language Modeling. In *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1995)*, pp. 181–184, Detroit, MI, USA, May 1995.

[Knight 99] K. Knight: Decoding Complexity in Word-Replacement Translation Models. *Computational Linguistics*, Vol. 25, No. 4, pp. 607–615, December 1999.

[Knight & Nair^{+} 06] K. Knight, A. Nair, N. Rathod, K. Yamada: Unsupervised Analysis for Decipherment Problems. In *Proceedings of the 2006 Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL 2006)*, pp. 499–506, Sydney, Austrailia, July 2006.

[Knight & Yamada 99] K. Knight, K. Yamada: A Computational Approach to Deciphering Unknown Scripts. In *Proceedings of the Workshop on Unsupervised Learning in Natural Language Processing*, pp. 37–44, College Park, MD, USA, June 1999.

[Kocmi & Bojar 18] T. Kocmi, O. Bojar: Trivial Transfer Learning for Low-Resource Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 244–252, Brussels, Belgium, November 2018.

[Koehn 05] P. Koehn: Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the 10th Machine Translation Summit (MT Summit X)*, pp. 79–86, Phuket, Thailand, September 2005.

[Koehn 10] P. Koehn: *Statistical Machine Translation.* Cambridge University Press, 2010.

[Koehn 20] P. Koehn: *Neural Machine Translation.* Cambridge University Press, 2020.

[Koehn & Guzmán^{+} 19] P. Koehn, F. Guzmán, V. Chaudhary, J. Pino: Findings of the WMT 2019 Shared Task on Parallel Corpus Filtering for Low-Resource Conditions. In *Proceedings of the Fourth Conference on Machine Translation (WMT 2019)*, pp. 54–72, Florence, Italy, July 2019.

[Koehn & Hoang 07a] P. Koehn, H. Hoang: Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pp. 868–876, Prague, Czech Republic, June 2007.

[Koehn & Hoang^{+} 07b] 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 Association for Computational Linguistics (ACL 2007)*, pp. 177–180, Prague, Czech Republic, June 2007.

[Koehn & Khayrallah$^+$ 18] P. Koehn, H. Khayrallah, K. Heafield, M.L. Forcada: Findings of the WMT 2018 Shared Task on Parallel Corpus Filtering. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 726–739, Brussels, Belgium, October 2018.

[Koehn & Knowles 17] P. Koehn, R. Knowles: Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation (NMT 2017)*, pp. 28–39, Vancouver, Canada, August 2017.

[Koehn & Monz 06] P. Koehn, C. Monz: Manual and Automatic Evaluation of Machine Translation between European Languages. In *Proceedings of the Workshop on Statistical Machine Translation (WMT 2006)*, pp. 102–121, New York City, NY, USA, June 2006.

[Koehn & Och$^+$ 03] P. Koehn, F.J. Och, D. Marcu: Statistical Phrase-Based Translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2003)*, pp. 48–54, Edmonton, Canada, May 2003.

[Lafferty & McCallum$^+$ 01] J.D. Lafferty, A. McCallum, F.C. Pereira: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pp. 282–289, Williamstown, MA, USA, June 2001.

[Lakew & Lotito$^+$ 17] S.M. Lakew, Q.F. Lotito, M. Negri, M. Turchi, M. Federico: Improving Zero-Shot Translation of Low-Resource Languages. In *Proceedings of the 14th International Workshop on Spoken Language Translation (IWSLT 2017)*, Tokyo, Japan, December 2017.

[Lample & Denoyer$^+$ 18] G. Lample, L. Denoyer, M. Ranzato: Unsupervised Machine Translation Using Monolingual Corpora Only. In *Proceedings of Sixth International Conference on Learning Representations (ICLR 2018)*, Vancouver, Canada, May 2018.

[Lample & Ott$^+$ 18] G. Lample, M. Ott, A. Conneau, L. Denoyer, M. Ranzato: Phrase-Based & Neural Unsupervised Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pp. 5039–5049, Brussels, Belgium, November 2018.

[Läubli & Sennrich$^+$ 18] S. Läubli, R. Sennrich, M. Volk: Has Machine Translation Achieved Human Parity? A Case for Document-level Evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pp. 4791–4796, Brussels, Belgium, November 2018.

[Le & Allauzen$^+$ 10] H.S. Le, A. Allauzen, G. Wisniewski, F. Yvon: Training Continuous Space Language Models: Some Practical Issues. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pp. 778–788, Cambridge, MA, USA, October 2010.

[Lee & Cho$^+$ 17] J. Lee, K. Cho, T. Hofmann: Fully Character-Level Neural Machine Translation without Explicit Segmentation. *Transactions of the Association for Computational Linguistics (TACL)*, Vol. 5, pp. 365–378, 2017.

[Lu & Keung$^+$ 18] Y. Lu, P. Keung, F. Ladhak, V. Bhardwaj, S. Zhang, J. Sun: A Neural Interlingua for Multilingual Machine Translation. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 84–92, Brussels, Belgium, November 2018.

[Luong & Le$^+$ 16] M.T. Luong, Q.V. Le, I. Sutskever, O. Vinyals, L.u. Kaiser: Multi-Task Sequence to Sequence Learning. In *Proceedings of the Fourth International Conference on Learning Representations (ICLR 2016)*, San Juan, Puerto Rico, May 2016.

[Luong & Manning 15] M.T. Luong, C.D. Manning: Stanford Neural Machine Translation Systems for Spoken Language Domains. In *Proceedings of the 12th International Workshop on Spoken Language Translation (IWSLT 2015)*, pp. 76–79, Da Nang, Vietnam, December 2015.

[Luong & Pham$^+$ 15] T. Luong, H. Pham, C.D. Manning: Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pp. 1412–1421, Lisbon, Portugal, September 2015.

[Manning & Surdeanu$^+$ 14] C.D. Manning, M. Surdeanu, J. Bauer, J.R. Finkel, S. Bethard, D. McClosky: The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pp. 55–60, Baltimore, MD, USA, June 2014.

[Marcu & Wong 02] D. Marcu, W. Wong: A Phrase-based, Joint Probability Model for Statistical Machine Translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pp. 133–139, Philadelphia, PA, USA, July 2002.

[Marie & Sun$^+$ 19] B. Marie, H. Sun, R. Wang, K. Chen, A. Fujita, M. Utiyama, E. Sumita: NICT's Unsupervised Neural and Statistical Machine Translation Systems for the WMT19 News Translation Task. In *Proceedings of the Fourth Conference on Machine Translation (WMT 2019)*, pp. 294–301, Florence, Italy, August 2019.

[Martin & Liermann$^+$ 98] S. Martin, J. Liermann, H. Ney: Algorithms for Bigram and Trigram Word Clustering. *Speech Communication*, Vol. 24, No. 1, pp. 19–37, April 1998.

[Mauser & Hasan$^+$ 09] A. Mauser, S. Hasan, H. Ney: Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pp. 210–218, Singapore, August 2009.

[Mays & Damerau$^+$ 91] E. Mays, F.J. Damerau, R.L. Mercer: Context Based Spelling Correction. *Information Processing & Management*, Vol. 27, No. 5, pp. 517–522, 1991.

[McCloskey & Cohen 89] M. McCloskey, N.J. Cohen: Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of Learning and Motivation*, Vol. 24, pp. 109–165. Elsevier, 1989.

[Mikolov & Chen$^+$ 13] T. Mikolov, K. Chen, G. Corrado, J. Dean: Efficient Estimation of Word Representations in Vector Space. arXiv cs.CL 1301.3781, 2013.

[Mikolov & Karafiát$^+$ 10] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, S. Khudanpur: Recurrent Neural Network Based Language Model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, pp. 1045–1048, Chiba, Japan, September 2010.

[Mikolov & Le$^+$ 13] T. Mikolov, Q.V. Le, I. Sutskever: Exploiting Similarities among Languages for Machine Translation. arXiv cs.CL 1309.4168, 2013.

[Mikolov & Sutskever$^+$ 13] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean: Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing System 26 (NIPS 2013)*, pp. 3111–3119, Lake Tahoe, NV, USA, December 2013.

[Miura & Neubig$^+$ 15] A. Miura, G. Neubig, S. Sakti, T. Toda, S. Nakamura: Improving Pivot Translation by Remembering the Pivot. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the Seventh International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, Vol. 2, pp. 573–577, Beijing, China, July 2015.

[Mnih & Kavukcuoglu 13] A. Mnih, K. Kavukcuoglu: Learning Word Embeddings Efficiently with Noise-contrastive Estimation. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pp. 2265–2273, Lake Tahoe, NV, USA, December 2013.

[Mnih & Teh 12] A. Mnih, Y.W. Teh: A Fast and Simple Algorithm for Training Neural Probabilistic Language Models. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, pp. 419–426, Edinburgh, Scotland, June 2012.

[Mohamed & Dahl$^+$ 12] A. Mohamed, G.E. Dahl, G. Hinton: Acoustic Modeling Using Deep Belief Networks. *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 20, No. 1, pp. 14–22, January 2012.

[Moore 02] R.C. Moore: Fast and Accurate Sentence Alignment of Bilingual Corpora. In *Proceedings of the Fifth Conference of the Association for Machine Translation in the Americas (AMTA 2002)*, pp. 135–144, Tiburon, CA, USA, October 2002.

[Naim & Riley$^+$ 18] I. Naim, P. Riley, D. Gildea: Feature-Based Decipherment for Machine Translation. *Computational Linguistics*, Vol. 44, No. 3, pp. 525–546, 2018.

[Nakashole & Flauger 18] N. Nakashole, R. Flauger: Characterizing Departures from Linearity in Word Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pp. 221–227, Melbourne, Austrailia, July 2018.

[Nalisnick & Mitra$^+$ 16] E. Nalisnick, B. Mitra, N. Craswell, R. Caruana: Improving Document Ranking with Dual Word Embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web (WWW 2016)*, pp. 83–84, Montreal, Canada, April 2016.

[Neubig & Hu 18] G. Neubig, J. Hu: Rapid Adaptation of Neural Machine Translation to New Languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pp. 875–880, Brussels, Belgium, October 2018.

[Nguyen & Chiang 17] T.Q. Nguyen, D. Chiang: Transfer Learning across Low-Resource, Related Languages for Neural Machine Translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP 2017)*, pp. 296–301, Taipei, Taiwan, November 2017.

[Nirenburg 89] S. Nirenburg: Knowledge-Based Machine Translation. *Machine Translation*, Vol. 4, No. 1, pp. 5–24, 1989.

[Nix 19] A. Nix: Multitask Learning for Neural Machine Translation. Master's thesis, Department of Computer Science, RWTH Aachen University, Aachen, Germany, 2019.

[Nuhn 19] M. Nuhn: *Unsupervised Training with Applications in Natural Language Processing.* Ph.D. thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, July 2019.

[Nuhn & Mauser$^+$ 12] M. Nuhn, A. Mauser, H. Ney: Deciphering Foreign Language by Combining Language Models and Context Vectors. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pp. 156–164, Jeju, Republic of Korea, July 2012.

[Nuhn & Ney 14] M. Nuhn, H. Ney: EM Decipherment for Large Vocabularies. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pp. 759–764, Baltimore, MD, USA, June 2014.

[Nuhn & Schamper$^+$ 13] M. Nuhn, J. Schamper, H. Ney: Beam Search for Solving Substitution Ciphers. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pp. 1569–1576, Sofia, Bulgaria, August 2013.

[Och 03] F.J. Och: Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pp. 160–167, Sapporo, Japan, July 2003.

[Och & Ney 02] F.J. Och, H. Ney: Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pp. 295–302, Philadelphia, PA, USA, July 2002.

[Och & Ney 03] F.J. Och, H. Ney: A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, Vol. 29, No. 1, pp. 19–51, March 2003.

[Olson & Wyner$^+$ 18] M. Olson, A. Wyner, R. Berk: Modern Neural Networks Generalize on Small Data Sets. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pp. 3619–3628, Montréal, Canada, December 2018.

[Ott & Auli$^+$ 18] M. Ott, M. Auli, D. Grangier, M. Ranzato: Analyzing Uncertainty in Neural Machine Translation. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pp. 3956–3965, Stockholm, Sweden, July 2018.

[Papineni & Roukos$^+$ 02] K. Papineni, S. Roukos, T. Ward, W.J. Zhu: BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pp. 311–318, Philadelphia, PA, USA, July 2002.

[Pascanu & Mikolov$^+$ 13] R. Pascanu, T. Mikolov, Y. Bengio: On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, pp. 1310–1318, Atlanta, GA, USA, June 2013.

[Paszke & Gross$^+$ 19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pp. 8026–8037, Vancouver, Canada, December 2019.

[Peirsman & Padó 10] Y. Peirsman, S. Padó: Cross-lingual Induction of Selectional Preferences with Bilingual Vector Spaces. In *Proceedings of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2010)*, pp. 921–929, Los Angeles, CA, USA, June 2010.

[Peter & Beck$^+$ 18] J.T. Peter, E. Beck, H. Ney: Sisyphus, a Workflow Manager Designed for Machine Translation and Automatic Speech Recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pp. 84–89, Brussels, Belgium, November 2018.

[Peter & Nix$^+$ 17] J.T. Peter, A. Nix, H. Ney: Generating Alignments Using Target Foresight in Attention-based Neural Machine Translation. *The Prague Bulletin of Mathematical Linguistics*, Vol. 108, No. 1, pp. 27–36, 2017.

[Peters & Neumann$^+$ 18] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer: Deep Contextualized Word Representations. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018)*, pp. 2227–2237, New Orleans, LA, USA, June 2018.

[Peters & Ruder$^+$ 19] M.E. Peters, S. Ruder, N.A. Smith: To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. In *Proceedings of the Fourth Workshop on Representation Learning for NLP (RepL4NLP 2019)*, pp. 7–14, Florence, Italy, July 2019.

[Petrov 19] P. Petrov: Neural Machine Translation between Non-English Languages. Master's thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, September 2019.

[Pires & Schlinger$^+$ 19] T. Pires, E. Schlinger, D. Garrette: How Multilingual is Multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pp. 4996–5001, Florence, Italy, July 2019.

[Platanios & Sachan$^+$ 18] E.A. Platanios, M. Sachan, G. Neubig, T. Mitchell: Contextual Parameter Generation for Universal Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pp. 425–435, Brussels, Belgium, October 2018.

[Post 18] M. Post: A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 186–191, Brussels, Belgium, November 2018.

[Press & Wolf 17] O. Press, L. Wolf: Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pp. 157–163, Valencia, Spain, April 2017.

[Qi & Sachan$^+$ 18] Y. Qi, D. Sachan, M. Felix, S. Padmanabhan, G. Neubig: When and Why Are Pre-Trained Word Embeddings Useful for Neural Machine Translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018)*, pp. 529–535, 2018.

[Radford & Narasimhan$^+$ 18] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever: Improving Language Understanding by Generative Pre-Training. Technical report, OpenAI, June 2018.

[Ramachandran & Liu$^+$ 17] P. Ramachandran, P. Liu, Q. Le: Unsupervised Pretraining for Sequence to Sequence Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pp. 383–391, Copenhagen, Denmark, September 2017.

[Ranzato & Chopra$^+$ 16] M. Ranzato, S. Chopra, M. Auli, W. Zaremba: Sequence Level Training with Recurrent Neural Networks. arXiv cs.LG 1511.06732, 2016.

[Ratcliff 90] R. Ratcliff: Connectionist Models of Recognition Memory: Constraints Imposed by Learning and Forgetting Functions. *Psychological Review*, Vol. 97, No. 2, pp. 285, 1990.

[Ratnaparkhi 96] A. Ratnaparkhi: A Maximum Entropy Model for Part-of-Speech Tagging. In *Proceedings of the 1996 Conference on Empirical Methods in Natural Language Processing (EMNLP 1996)*, pp. 133–142, Philadelphia, PA, USA, May 1996.

[Ravi 13] S. Ravi: Scalable Decipherment for Machine Translation via Hash Sampling. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pp. 362–371, Sofia, Bulgaria, August 2013.

[Ravi & Knight 11a] S. Ravi, K. Knight: Bayesian Inference for Zodiac and Other Homophonic Ciphers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pp. 19–24, Portland, OR, USA, June 2011.

[Ravi & Knight 11b] S. Ravi, K. Knight: Deciphering Foreign Language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pp. 12–21, Portland, OR, USA, June 2011.

[Ren & Chen$^+$ 18] S. Ren, W. Chen, S. Liu, M. Li, M. Zhou, S. Ma: Triangular Architecture for Rare Language Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pp. 56–65, Melbourne, Austrailia, July 2018.

[Ren & Wu$^+$ 19] S. Ren, Y. Wu, S. Liu, M. Zhou, S. Ma: Explicit Cross-lingual Pre-training for Unsupervised Machine Translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the Ninth International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*, pp. 770–779, Hong Kong, China, November 2019.

[Ren & Zhang$^+$ 19] S. Ren, Z. Zhang, S. Liu, M. Zhou, S. Ma: Unsupervised Neural Machine Translation with SMT as Posterior Regularization. arXiv cs.CL 1901.04112, 2019.

[Resnik & Smith 03] P. Resnik, N.A. Smith: The Web as a Parallel Corpus. *Computational Linguistics*, Vol. 29, No. 3, pp. 349–380, 2003.

[Riley & Gildea 18] P. Riley, D. Gildea: Orthographic features for bilingual lexicon induction. In *ACL*, pp. 390–394, 2018.

[Rishøj & Søgaard 11] C. Rishøj, A. Søgaard: Factored Translation with Unsupervised Word Clusters. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT 2011)*, pp. 447–451, Edinburgh, Scotland, July 2011.

[Rosendahl & Herold$^+$ 19] J. Rosendahl, C. Herold, Y. Kim, M. Graça, W. Wang, P. Bahar, Y. Gao, H. Ney: The RWTH Aachen University Machine Translation Systems for WMT 2019. In *Proceedings of the Fourth Conference on Machine Translation (WMT 2019)*, pp. 349–355, Florence, Italy, August 2019.

[Rossenbach & Rosendahl$^+$ 18] N. Rossenbach, J. Rosendahl, Y. Kim, M. Graça, A. Gokrani, H. Ney: The RWTH Aachen University Filtering System for the WMT 2018 Parallel Corpus Filtering Task. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 946–954, Belgium, Brussels, October 2018.

[Ruder & Vulić$^+$ 19] S. Ruder, I. Vulić, A. Søgaard: A Survey of Cross-lingual Word Embedding Models. *Journal of Artificial Intelligence Research*, Vol. 65, pp. 569–631, 2019.

[Rumelhart & Hinton$^+$ 86] D.E. Rumelhart, G.E. Hinton, R.J. Williams: Learning Representations by Back-Propagating Errors. *Nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

[Sachan & Neubig 18] D. Sachan, G. Neubig: Parameter Sharing Methods for Multilingual Self-Attentional Translation Models. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 261–271, Brussels, Belgium, October 2018.

[Schamper 15] J. Schamper: Unsupervised Training with Applications in Natural Language Processing. Master's thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, September 2015.

[Schamper & Rosendahl$^+$ 18] J. Schamper, J. Rosendahl, P. Bahar, Y. Kim, A. Nix, H. Ney: The RWTH Aachen University Supervised Machine Translation Systems for WMT 2018. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 496–503, Belgium, Brussels, October 2018.

[Schwarz 78] G. Schwarz: Estimating the Dimension of a Model. *The Annals of Statistics*, Vol. 6, No. 2, pp. 461–464, 1978.

[Schwenk 08] H. Schwenk: Investigations on Large-Scale Lightly-supervised Training for Statistical Machine Translation. In *Proceedings of the Fifth International Workshop on Spoken Language Translation (IWSLT 2008)*, pp. 182–189, Honolulu, HI, USA, October 2008.

[Schwenk 12] H. Schwenk: Continuous Space Translation Models for Phrase-Based Statistical Machine Translation. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pp. 1071–1080, Mumbai, India, December 2012.

[Schwenk & Chaudhary$^+$ 19] H. Schwenk, V. Chaudhary, S. Sun, H. Gong, F. Guzmán: WikiMatrix: Mining 135M Parallel Sentences in 1620 Language Pairs from Wikipedia. arXiv cs.CL 1907.05791, 2019.

[Schwenk & Déchelotte$^+$ 06] H. Schwenk, D. Déchelotte, J.L. Gauvain: Continuous Space Language Models for Statistical Machine Translation. In *Proceedings of the 21th International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*, pp. 723–730, Sydney, Austrailia, July 2006.

[Schwenk & Rousseau$^+$ 12] H. Schwenk, A. Rousseau, M. Attik: Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pp. 11–19, Montreal, Canada, June 2012.

[Sen & Gupta$^+$ 19] S. Sen, K.K. Gupta, A. Ekbal, P. Bhattacharyya: Multilingual Unsupervised NMT using Shared Encoder and Language-Specific Decoders. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pp. 3083–3089, Florence, Italy, July 2019.

[Sennrich & Birch$^+$ 17] R. Sennrich, A. Birch, A. Currey, U. Germann, B. Haddow, K. Heafield, A.V. Miceli-Barone, P. Williams: The University of Edinburgh's Neural MT Systems for WMT17. In *Proceedings of the Second Conference on Machine Translation (WMT 2017)*, pp. 389–399, Copenhagen, Denmark, September 2017.

[Sennrich & Haddow$^+$ 16a] R. Sennrich, B. Haddow, A. Birch: Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation (WMT 2016)*, pp. 371–376, Berlin, Germany, August 2016.

[Sennrich & Haddow$^+$ 16b] R. Sennrich, B. Haddow, A. Birch: Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pp. 86–96, Berlin, Germany, July 2016.

[Sennrich & Haddow$^+$ 16c] R. Sennrich, B. Haddow, A. Birch: Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pp. 1715–1725, Berlin, Germany, August 2016.

[Sennrich & Zhang 19] R. Sennrich, B. Zhang: Revisiting Low-Resource Neural Machine Translation: A Case Study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pp. 211–221, Florence, Italy, July 2019.

[Sestorain & Ciaramita$^+$ 18] L. Sestorain, M. Ciaramita, C. Buck, T. Hofmann: Zero-Shot Dual Machine Translation. arXiv cs.CL 1805.10338, 2018.

[Shannon 48] C.E. Shannon: A Mathematical Theory of Communication. *Bell System Technical Journal*, Vol. 27, No. 3, pp. 379–423, 1948.

[Siddhant & Bapna$^+$ 20] A. Siddhant, A. Bapna, Y. Cao, O. Firat, M.X. Chen, S. Kudugunta, N. Arivazhagan, Y. Wu: Leveraging Monolingual Data with Self-Supervision for Multilingual Neural Machine Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pp. 2827–2835, Virtual Conference, July 2020.

[Smith & Saint-Amand$^+$ 13] J.R. Smith, H. Saint-Amand, M. Plamada, P. Koehn, C. Callison-Burch, A. Lopez: Dirt Cheap Web-Scale Parallel Text from the Common Crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pp. 1374–1383, Sofia, Bulgaria, August 2013.

[Smith & Turban$^+$ 17] S.L. Smith, D.H.P. Turban, S. Hamblin, N.Y. Hammerla: Offline Bilingual Word Vectors, Orthogonal Transformations and the Inverted Softmax. In *Proceedings of Sixth International Conference on Learning Representations (ICLR 2017)*, Toulon, France, April 2017.

[Snover & Dorr$^+$ 06] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul: A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the Seventh Conference of the Association for Machine Translation in the Americas (AMTA 2006)*, pp. 223–231, Cambridge, MA, USA, August 2006.

[Søgaard & Ruder$^+$ 18] A. Søgaard, S. Ruder, I. Vulić: On the Limitations of Unsupervised Bilingual Dictionary Induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pp. 778–788, Melbourne, Austrailia, July 2018.

[Son & Allauzen$^+$ 12] L.H. Son, A. Allauzen, F. Yvon: Continuous Space Translation Models with Neural Networks. In *Proceedings of the 12th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2012)*, pp. 39–48, Montreal, Canada, June 2012.

[Song & Tan$^+$ 19] K. Song, X. Tan, T. Qin, J. Lu, T.Y. Liu: MASS: Masked Sequence to Sequence Pre-training for Language Generation. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, pp. 5926–5936, Long Beach, CA, USA, June 2019.

[Sriram & Jun$^+$ 18] A. Sriram, H. Jun, S. Satheesh, A. Coates: Cold Fusion: Training Seq2Seq Models Together with Language Models. In *Proceedings of the 19th Annual Conference of the International Speech Communication Association (INTERSPEECH 2018)*, pp. 387–391, Hyderabad, India, September 2018.

[Srivastava & Hinton$^+$ 14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, Vol. 15, No. 1, pp. 1929–1958, 2014.

[Stahlberg & Cross$^+$ 18] F. Stahlberg, J. Cross, V. Stoyanov: Simple Fusion: Return of the Language Model. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 204–211, Brussels, Belgium, November 2018.

[Sun & Wang$^+$ 19] H. Sun, R. Wang, K. Chen, M. Utiyama, E. Sumita, T. Zhao: Unsupervised Bilingual Word Embedding Agreement for Unsupervised Neural Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pp. 1235–1245, Florence, Italy, July 2019.

[Sundermeyer & Alkhouli$^+$ 14] M. Sundermeyer, T. Alkhouli, J. Wuebker, H. Ney: Translation Modeling with Bidirectional Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pp. 14–25, Doha, Qatar, October 2014.

[Sutskever & Vinyals$^+$ 14] I. Sutskever, O. Vinyals, Q.V. Le: Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pp. 3104–3112, Montréal, Canada, December 2014.

[Szegedy & Vanhoucke$^+$ 16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna: Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pp. 2818–2826, Las Vegas, NV, USA, June 2016.

[Taghipour & Khadivi$^+$ 11] K. Taghipour, S. Khadivi, J. Xu: Parallel Corpus Refinement as an Outlier Detection Algorithm. In *Proceedings of the 13th Machine Translation Summit (MT Summit XIII)*, pp. 414–421, Xiamen, China, September 2011.

[Tang & Müller$^+$ 18] G. Tang, M. Müller, A. Rios, R. Sennrich: Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pp. 4263–4272, Brussels, Belgium, November 2018.

[Taylor 53] W.L. Taylor: Cloze Procedure: A New Tool for Measuring Readability. *Journalism Quarterly*, Vol. 30, No. 4, pp. 415–433, 1953.

[Thrun & Pratt 12] S. Thrun, L. Pratt: *Learning to Learn.* Springer, December 2012.

[Tiedemann 12] J. Tiedemann: Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the Eighth Language Resources and Evaluation Conference (LREC 2012)*, pp. 2214–2218, Istanbul, Turkey, May 2012.

[Toral & Castilho$^+$ 18] A. Toral, S. Castilho, K. Hu, A. Way: Attaining the Unattainable? Reassessing Claims of Human Parity in Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pp. 113–123, Brussels, Belgium, November 2018.

[Tran & Bisk$^+$ 16] K. Tran, Y. Bisk, A. Vaswani, D. Marcu, K. Knight: Unsupervised Neural Hidden Markov Models. In *Proceedings of the EMNLP 2016 Workshop on Structured Prediction for Natural Language Processing (SP4NLP 2016)*, Austin, TX, USA, November 2016.

[Ueffing 06] N. Ueffing: Using Monolingual Source-Language Data to Improve MT Performance. In *Proceedings of the Third International Workshop on Spoken Language Translation (IWSLT 2006)*, pp. 174–181, Kyoto, Japan, November 2006.

[Ueffing & Haffari$^+$ 07] N. Ueffing, G. Haffari, A. Sarkar: Transductive Learning for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pp. 25–32, Prague, Czech Republic, June 2007.

[Utiyama & Isahara 07] M. Utiyama, H. Isahara: A Comparison of Pivot Methods for Phrase-based Statistical Machine Translation. In *Proceedings of the Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2007)*, pp. 484–491, Rochester, NY, USA, April 2007.

[Vapnik 98] V.N. Vapnik: *Statistical Learning Theory.* Wiley, Hoboken, NJ, USA, 1998.

[Varga & Kornai$^+$ 05] D. Varga, A. Kornai, V. Nagy, L. Németh, V. Trón: Parallel Corpora for Medium Density Languages. In *Proceedings of the Fifth International Conference on Recent Advances in Natural Language Processing (RANLP 2005)*, pp. 590–596, Borovets, Bulgaria, September 2005.

[Vaswani & Shazeer$^+$ 17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. ukasz Kaiser, I. Polosukhin: Attention is All You Need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pp. 5998–6008, Long Beach, CA, USA, December 2017.

[Vauquois 68] B. Vauquois: A Survey of Formal Grammars and Algorithms for Recognition and Transformation in Machine Translation. In *Proceedings of the International Federation for Information Processing Congress 1968 (IFIP Congress 1968)*, pp. 1114–1122, Edinburgh, UK, August 1968.

[Vilar & Stein$^+$ 10] D. Vilar, D. Stein, M. Huck, H. Ney: Jane: Open Source Hierarchical Translation, Extended with Reordering and Lexicon Models. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR (WMT 2010)*, pp. 262–270, Uppsala, Sweden, July 2010.

[Vincent & Larochelle$^+$ 08] P. Vincent, H. Larochelle, Y. Bengio, P.A. Manzagol: Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pp. 1096–1103, Helsinki, Finland, July 2008.

[Vincent & Larochelle$^+$ 10] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.A. Manzagol, L. Bottou: Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research (JMLR)*, Vol. 11, No. 12, pp. 3371–3408, 2010.

[Voita & Sennrich$^+$ 19] E. Voita, R. Sennrich, I. Titov: The Bottom-up Evolution of Representations in the Transformer: A Study with Machine Translation and Language Modeling Objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the Ninth International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*, pp. 4387–4397, Hong Kong, China, November 2019.

[Voita & Talbot$^+$ 19] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, I. Titov: Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pp. 5797–5808, Florence, Italy, July 2019.

[Wada & Iwata 18] T. Wada, T. Iwata: Unsupervised Cross-lingual Word Embedding by Multilingual Neural Language Models. arXiv cs.CL 1809.02306, 2018.

[Wang & Pham$^+$ 19] X. Wang, H. Pham, P. Arthur, G. Neubig: Multilingual Neural Machine Translation With Soft Decoupled Encoding. In *Proceedings of the 2019 International Conference on Learning Representations (ICLR 2019)*, 2019.

[Way 13] A. Way: Traditional and Emerging Use-Cases for Machine Translation. In *Proceedings of the Translating and the Computer Conference 35 (TC35)*, London, UK, November 2013.

[Wu & Schuster$^+$ 16] 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, L. ukasz 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, J. Dean: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv cs.CL 1609.08144, 2016.

[Wu & Wang 07] H. Wu, H. Wang: Pivot Language Approach for Phrase-Based Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pp. 856–863, Prague, Czech Republic, June 2007.

[Wuebker & Peitz$^+$ 13] J. Wuebker, S. Peitz, F. Rietig, H. Ney: Improving Statistical Machine Translation with Word Class Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pp. 1377–1381, Seattle, USA, October 2013.

[Xing & Wang$^+$ 15] C. Xing, D. Wang, C. Liu, Y. Lin: Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation. In *Proceedings of the 14th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2015)*, pp. 1006–1011, Denver, CO, USA, June 2015.

[Xu & Koehn 17] H. Xu, P. Koehn: Zipporah: a Fast and Scalable Data Cleaning System for Noisy Web-Crawled Parallel Corpora. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pp. 2945–2950, Copenhagen, Denmark, September 2017.

[Xu & Niu$^+$ 20] W. Xu, X. Niu, M. Carpuat: Dual Reconstruction: a Unifying Objective for Semi-Supervised Neural Machine Translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pp. 2006–2020, Virtual Conference, November 2020.

[Yang & Chen$^+$ 18] Z. Yang, W. Chen, F. Wang, B. Xu: Unsupervised Neural Machine Translation with Weight Sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pp. 46–55, Melbourne, Austrailia, July 2018.

[Yee & Dauphin$^+$ 19] K. Yee, Y. Dauphin, M. Auli: Simple and Effective Noisy Channel Modeling for Neural Machine Translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the Ninth International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*, pp. 5700–5705, Hong Kong, China, November 2019.

[Zahabi & Bakhshaei$^+$ 13] S.T. Zahabi, S. Bakhshaei, S. Khadivi: Using Context Vectors in Improving a Machine Translation System with Bridge Language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pp. 318–322, Sofia, Bulgaria, August 2013.

[Zeiler 12] M.D. Zeiler: ADADELTA: An Adaptive Learning Rate Method. arXiv cs.LG 1212.5701, 2012.

[Zens & Ney 03] R. Zens, H. Ney: A Comparative Study on Reordering Constraints in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pp. 144–151, Sapporo, Japan, July 2003.

[Zens & Och$^+$ 02] R. Zens, F.J. Och, H. Ney: Phrase-Based Statistical Machine Translation. In M. Jarke, J. Koehler, G. Lakemeyer, editors, *Proceedings of the 25th German Conference on Artificial Intelligence (KI 2002)*, Vol. 2479 of *Lecture Notes in Artificial Intelligence (LNAI)*, pp. 18–32, Aachen, Germany, September 2002. Springer Verlag.

[Zeyer & Doetsch$^+$ 17] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlüter, H. Ney: A Comprehensive Study of Deep Bidirectional LSTM RNNs for Acoustic Modeling in Speech Recognition. In *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*, pp. 2462–2466, New Orleans, LA, USA, March 2017.

[Zhang & Liu$^+$ 17] M. Zhang, Y. Liu, H. Luan, M. Sun: Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pp. 1959–1970, Vancouver, Canada, July 2017.

[Zheng & Cheng$^+$ 17] H. Zheng, Y. Cheng, Y. Liu: Maximum Expected Likelihood Estimation for Zero-Resource Neural Machine Translation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pp. 4251–4257, Melbourne, Australia, August 2017.

[Zhu & He$^+$ 14] X. Zhu, Z. He, H. Wu, C. Zhu, H. Wang, T. Zhao: Improving Pivot-based Statistical Machine Translation by Pivoting the Co-Occurrence Count of Phrase Pairs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pp. 1665–1675, Doha, Qatar, October 2014.

[Ziemski & Junczys-Dowmunt$^+$ 16] M. Ziemski, M. Junczys-Dowmunt, B. Pouliquen: The United Nations Parallel Corpus v1.0. In *Proceedings of the 10th Language Resources and Evaluation Conference (LREC 2016)*, pp. 3530–3534, Portorož, Slovenia, May 2016.

[Zoph & Yuret$^+$ 16] B. Zoph, D. Yuret, J. May, K. Knight: Transfer Learning for Low-Resource Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pp. 1568–1575, Austin, TX, USA, November 2016.