# Attention-based Machine Translation Using Monolingual Data

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.**
**Jan Rosendahl**
aus Köln, Deutschland

Berichter: Universitätsprofessor Dr.-Ing. Hermann Ney
Universitätsprofessor Dr. Francisco Casacuberta

Tag der mündlichen Prüfung: 22. Februar 2024

# Acknowledgments

I am grateful to many people who helped me in so many different ways during my PhD journey.

First and foremost, I would like to express my gratitude to Professor Dr-Ing. Hermann Ney, for offering me the great opportunity of working at his chair, for teaching me how to perform state-of-the-art research, and for opening so many doors to me. His dedication to the problem, his intense knowledge, and his attention to detail were truly inspiring. I would also like to thank Professor Dr. Francisco Casacuberta for accepting to be my second supervisor, for his valuable feedback and his profound interest in my dissertation.

It cannot be overstated how important my colleagues at the institute were in the process of this work. They helped with great feedback, with in-depth discussions (whether they occurred in the university or a bar) and always increased my understanding. Jan-Thorsten Peter extended the world of machine learning to me and introduced me not only to machine translation and the technical workings of the institute but also to so many researchers across the international community. Together with Julian Schamper I dove into machine learning theory, overhauled a lecture, and started some ambitious research projects. Some of these projects I ended up concluding with Yunsu Kim who impressed me with his calm approach and helped me to face difficult decisions. With Christian Herold, I dug into data filtering as well as a variety of other topics and enjoyed more than one conference. With Parnia Bahar I had many conversations about the inner workings of neural models during our paper club. Brevity demands that I shorten some of my thanks but I want to particularly mention the other members of the machine translation group: David Thulke, Weiyue Wang, Ringo Gao, Tamer Alkhouli, and Andreas Guta. Thanks to all of you for your genuine interest, for your insights, for MT dinners and sports events. I could not have done it without you and it would have been way less fun.

I would like to thank my colleagues who became friends. Wilfried Michel and Christoph Luescher for clever and stupid talks as well as help in all situations. Thank you Pavel Golik for introducing me to the duties of a system administrator. This also set me up to get to know Stefan Koltermann and Thomas Dackweiler who were great colleagues, the perfect mix of stability and innovation and providers of fresh perspectives. Thanks to Peter Vieting for having me as an office mate on the last meters and for the good times we had. I was extremely fortunate to supervise some of the nicest and most talented students. Their work significantly influenced the directions of this dissertation: Frithjof Petrick, Philip Wynand, Joris Vanvinckenroye, Arne Nix, Evgeniia Tokarchuk, Nick Rossenbach, Lena Verscht, Hannah Mertens, Khoa Tran and Aman Gokrani. I supervised many of these students with Sharam Khadivi, who always found time in his calendar and week after week found helpful extensions to our work. A special thanks to Steffi Jansen who keeps an overview of everything that is going on at the institute and navigates the pitfalls of the academic bureaucracy.

Lastly, I want to express my deepest gratitude towards my girlfriend Nina, and my family in general, who has always supported me.

A PhD is a long process with highs and lows and through all these times you were with me. Thank you all.

# Abstract

Neural networks present a major advance in modeling for statistical machine translation systems. These data-driven systems consist of an encoder that computes a representation of the source sentence and a decoder that accesses the encoder output and generates a probability distribution over all target sentences. The components are connected via a cross-attention layer and trained jointly to minimize the cross-entropy loss on a corpus of bilingual training data, i.e. a set of sentence pairs where one is the translation of the other. In this dissertation, we focus on two important aspects of neural machine translation systems, namely the training data and the attention layer.

Since sentence-aligned bilingual data is a scarce resource and availability depends on the language pair, we investigate the use of monolingual data to improve the performance of the machine translation system. We verify the results reported with the use of synthetic data (back-translation) and extended language model fusion and introduce pre-training to neural machine translation. Using a language model trained on monolingual target data is an established method in count-based machine translation approaches. We adapt this to neural machine translation and extend the approach by training the parameters of the translation model as part of a greater fusion model. Furthermore, we use monolingual source and target data to find a better initialization for the training. This pre-training also allows the use of monolingual source data, which is commonly ignored in machine translation systems. We evaluate these methods empirically on four different language pairs with different data conditions and report improvements for all described methods over a purely bilingual baseline. Overall, back-translation provides the best results with respect to translation performance and data efficiency.

Inspired by existing work on alignment models, we also incorporate a first-order dependency in the attention layer. In contrast with previous machine translation models, the transformer is a purely feed-forward model without any recurrent layers. This means that no information about the previous attention decision is input to the computation of the attention layer. Modeling attention with first-order dependencies allows the attention layer to access previous attention decisions, which is an important prerequisite to express, e.g. source coverage. We adapt and propose several extensions to include this time-dependent information. Interpreting attention as a soft-lookup of a query to a list of key-value pairs, we introduce the previous attention information in different ways and using different encodings. All methods are verified on several machine translation tasks and we conclude that a zero-order attention model is sufficiently strong for the task of machine translation.

# Kurzfassung

Künstliche neuronale Netze stellen einen bedeutenden Fortschritt in der Modellierung von maschinellen Übersetzungssystemen mittels statistischen Methoden dar. Aktuelle maschinelle Übersetzungssysteme auf Basis neuronaler Netzwerke bestehen aus einem Encoder, der eine Repräsentation des Quellsatzes berechnet, und einem Decoder, der auf die Encoderausgabe zugreift um eine Wahrscheinlichkeitsverteilung über alle Zielsätze zu erzeugen. Diese beiden Komponenten sind mittels eines Cross-Attention Layers verbunden und werden gemeinsam trainiert. Dabei werden die Parameter des neuronalen Netzwerks so gewählt, dass die Verlustfunktion auf den zweisprachigen Trainingsdaten minimiert wird. Im Rahmen dieser Dissertation konzentrieren wir uns auf zwei zentrale Aspekte neuronaler Übersetzungssysteme: die verwendeten Trainingsdaten und das Cross-Attention Layer.

Da zweisprachige, satzweise-parallele Trainingsdaten eine knappe Ressource sind, deren Verfügbarkeit stark vom betrachteten Sprachpaar abhängt, untersuchen wir die Verwendung monolingualer Daten, also einsprachiger Sätze ohne zugeordnete Übersetzungen, um die Leistung eines maschinellen Übersetzungssystems zu verbessern. Wir verifizieren bestehende Ergebnisse, welche durch den Einsatz von synthetischen Daten (Rückwärtsübersetzung oder 'back-translation') und der Integration von Sprachmodellen erzielt wurden, darüber hinaus führen wir ein neues Vorabtraining für maschinelle Übersetzungssysteme ein. Die Verwendung eines Sprachmodells, das auf monolingualen Daten in der Zielsprache trainiert wurde, ist eine etablierte Methode in phrasen-basierten maschinellen Übersetzungsansätzen. Wir passen dieses Vorgehen an neuronale Übersetzungssysteme an und erweitern den Ansatz, indem wir die Parameter des Übersetzungsmodells als Teil eines fusionierten Modells trainieren. Des weiteren verwenden wir monolinguale Daten, um einen besseren Initialisierungspunkt für das Training zu finden. Dieses Vorabtraining ermöglicht unter anderem die Verwendung monolingualer Daten in der Quellsprache, welche üblicherweise bei der Verwendung maschineller Übersetzungssystemen ignoriert werden. Wir evaluieren alle beschriebenen Methoden empirisch anhand von vier verschiedenen Sprachpaaren mit unterschiedlichen Datenbedingungen. Alle Methoden erzielen bessere Ergebnisse als ein starkes Baseline-System, das im Training ausschließlich zweisprachige Daten verwendet. Wir beobachten, dass Rückwärtsübersetzung insgesamt die besten Ergebnisse liefert, sowohl in Bezug auf die Qualität des resultierenden Übersetzungssystems als auch im Bezug auf die Dateneffizienz.

Inspiriert von früheren Arbeiten im Bereich der Alignment-Modelle integrieren wir eine Abhängigkeit erster Ordnung in das Cross-Attention Layer. Aktuelle neuronale Übersetzungsmodelle sind reine Feed-Forward-Netzwerke, ohne rekurrente Layer. Das bedeutet, dass die Attention-Entscheidungen, die das aktuelle Zielwort generieren, keinen Einfluss auf die Attention-Entscheidung für Folgewörter haben. Die Modellierung eines Attention Layers mit Abhängigkeiten erster Ordnung ermöglicht es auf frühere Attention-Entscheidung zuzugreifen, was eine wichtige Voraussetzung ist, um z. B. auszudrücken welche Worte des Quellsatzes bereits durch die aktuelle Übersetzung abgedeckt sind. Im Rahmen dieser Arbeit adaptieren wir bestehende Erweiterungen des Cross-Attention Layers und schlagen mehrere neue vor, um diese zeitabhängige Information einzubeziehen. Alle Methoden werden auf verschiedenen Sprachpaaren geprüft und wir kommen zu dem Schluss, dass ein Attention-Modell ohne erweiterte Abhängigkeiten ausreichend ist.

# Contents

# 1. Introduction

The task of machine translation is to build an algorithm that transfers a sentence from a natural source language into a desired target language while preserving its meaning. Crucially, all steps of the process are executed by a computer program and do not require human intervention. To express the connection between the source and target language, historically two major types of tools are used: human-crafted rules and statistical models. As Koehn points out, these two approaches are frequently mixed and interconnected (See [Koehn 10, Sec 1.5.2 ]); however broadly speaking, rule-based translation systems require human experts on one or more languages to craft rules that either describe how a language is set up or how it is transformed into another language. In contrast to this, statistical methods are data-driven, meaning that a human-engineered probabilistic model is adapted to the task by optimizing its free parameters in such a way that a loss function is minimized on a corpus of language data. In this work we focus only on statistical approaches to the machine translation problem, which provide state-of-the-art results and are used in many commercial applications.

## 1.1 Statistical Machine Translation

Machine translation is considered a text-to-text problem, assuming both input and output sentences are represented as strings. Count-based approaches on word- [Brown & Cocke$^+$ 90] and phrase-level [Zens & Och$^+$ 02] proved that statistical machine translation systems are powerful tools capable of translating between a variety of languages [Ney 01, Koehn & Monz 06]. These data-driven systems require a corpus of bilingual training data, namely a set of pairs each consisting of a source sentence together with a translation. In an optimization process called *training* the free parameters of the stochastic model are chosen in such a way that they fit the available *training data*. Neural networks presented a major advancement in modeling for statistical machine translation systems [Sutskever & Vinyals$^+$ 14, Bahdanau & Cho$^+$ 15, Vaswani & Shazeer$^+$ 17]. Early neural machine translation (NMT) systems encode the source sentence into a single vector of fixed length [Sutskever & Vinyals$^+$ 14] using a long short-term memory (LSTM) layer [Hochreiter & Schmidhuber 97]. This vector is then decoded to a series of target words and the resulting encoder-decoder structure is used to this day in all neural machine translation systems. The introduction of an attention layer was a major advance for machine translation systems in two ways. First, a cross-attention layer was introduced to overcome the single vector bottleneck between the encoder and the decoder, which functions as a target-to-source soft-alignment [Bahdanau & Cho$^+$ 15]. In the second step, the expressiveness of the attention layers is increased via a multi-head attention mechanism, and the introduction of self-attention allows the replacement of the LSTM layers within the encoder and decoder [Vaswani & Shazeer$^+$ 17]. This current state-of-the-art architecture, introduced in 2017 as *transformer* [Vaswani & Shazeer$^+$ 17], yields superior performance across a variety of language pairs exemplified by the submissions and results of the prestigious shared task on machine translation by the Conference on Machine Translation

(WMT) [Kocmi & Bawden$^+$ 22]. Neural machine translation systems are commonly optimized to minimize the cross-entropy loss on a bilingual corpus. As statistical systems, they can only reproduce patterns that are observed during this training process. In this work, we focus on two central aspects of neural machine translation systems, namely the training data and the attention layer. Specifically, we increase the amount of training data by incorporating monolingual data, which is widely available, into the system and extend the attention layer to allow for first-order dependencies. In the following, we introduce both aspects of this investigation, discuss the scientific background and outline the proposed approaches.

## 1.2 Data in Statistical Machine Translation

Training data is extremely important for statistical systems in general and for neural machine translation in particular. Koehn and Knowles report that "NMT systems have a steeper learning curve with respect to the amount of training data, resulting in worse quality in low-resource settings, but better performance in high-resource settings" [Koehn & Knowles 17, p. 1]. They describe the amount of training data as one of six important challenges for the progression of machine translation systems. Another challenge is the handling of low-frequency words, which is directly tied to the amount of training data since the likelihood of unseen or rare words is increased if only small amounts of training data are available.

Sentence-aligned bilingual data is a scarce resource and the availability depends heavily on the language pair. Thus many approaches aim to increase or improve the training data of a machine translation system. This can be done by training on a resource-rich language to transfer the optimized parameters to a system for the intended language pair [Zoph & Yuret$^+$ 16] or by directly training a multilingual system that combines training data of up to a hundred languages [Johnson & Schuster$^+$ 17, Cettolo & Federico$^+$ 17, Aharoni & Johnson$^+$ 19]. Data augmentation techniques manipulate existing data to increase the diversity of the training signal either directly in the data [Fadaee & Bisazza$^+$ 17] or in the model representation of the data, e.g. via dropout [Srivastava & Hinton$^+$ 14]. Other approaches raise the amount of training data directly by accessing new resources, most commonly by the crawling of bilingual websites, which allows the extraction of several billion sentence pairs for many different language pairs [Bañón & Chen$^+$ 20]. A drawback of webcrawling is, that it commonly results in very noisy data which requires extensive data filtering [Rossenbach & Rosendahl$^+$ 18, Chaudhary & Tang$^+$ 19, Parcheta & Sanchis-Trilles$^+$ 19, Koehn & Chaudhary$^+$ 20].

Obtaining bilingual data is always challenging, whether it involves human experts translating, the automatic crawling of sentences from sources like the internet or navigating the legal requirements to access existing bilingual resources such as translated speeches or books. Most of these problems arise from the fact that machine translation systems require bilingual training data, which needs to be aligned as a sentence-by-sentence translation.

While obtaining bilingual data is problematic and resources are limited for a lot of language pairs, monolingual data is widely available for many languages. This raises the question of how to use this rich resource. Historically, in count-based, phrase-based machine translation systems monolingual data is used to train a separate language model that predicts the a priori probability of a word sequence [Zens & Och$^+$ 02]. In the current state-of-the-art method of back-translation [Schwenk 08, Sennrich & Haddow$^+$ 16b], the bilingual data is used to train a target-to-source translation system which is used to translate monolingual target data. This creates a synthetic bilingual corpus, that consists of human-written target sentences and automatically generated source sentences, which is added to the training data. In this work, we consider this state-of-art approach, verify its results and compare the performance to other approaches which include an external language model in training and search or use the monolingual data to train the parameters of the machine translation system.

When investigating the use of monolingual data it is important to consider the overall data situation. For certain language pairs no bilingual data is available, leading to the development of unsupervised machine translation systems that are trained exclusively on monolingual data [Artetxe & Labaka$^{+}$ 18]. Other works start with a tiny bilingual corpus which is insufficient to train a usable machine translation system but provides a starting point to add further resources [Chaudhary & Tang$^{+}$ 19]. In this work, we consider the case of a supervised task where enough bilingual data is present to build a machine translation with strong translation performance and focus on the question of how such a system can be improved using the vast amount of monolingual data available. We investigate a variety of data conditions with respect to the amount and domain of the bilingual and monolingual data and put a strong emphasis on a diversity of different approaches.

## 1.3 Alignment and Attention

Attention is an essential component of machine translation systems first applied to the task by Bahdanau et al. [Bahdanau & Cho$^{+}$ 15]. In the transformer architecture [Vaswani & Shazeer$^{+}$ 17], the current state-of-the-art [Akhbardeh & Arkhangorodsky$^{+}$ 21], attention is used as a sequence processing layer and as a connection between the encoder and the decoder. In this work, we focus on the *encoder-decoder attention* or *cross-attention*, which was originally introduced as a replacement for an alignment model [Bahdanau & Cho$^{+}$ 15]. For each target word a probability distribution over the words in the source sentence is generated, to weigh how much a certain source word ‘influences’ the current target word. This is a relaxation of a traditional target-to-source alignment model which maps each target word to a single source word and many works investigate the relationship between attention and alignment [Alkhouli & Bretschner$^{+}$ 16, Alkhouli & Bretschner$^{+}$ 18, Li & Li$^{+}$ 19, Zenkel & Wuebker$^{+}$ 20].

Crucially, the cross-attention layer is the only component that transports information about the source sentence into the decoder. Since it is the only connection between the decoder and the encoder, it is the only layer in which source and target information are merged. Obtaining the right information about the source sentence is a prerequisite for a good translation. Hence, in this work we consider the input to the cross-attention component, analyze which information is available and extend its dependencies. Previous works disagree on whether additional dependencies are beneficial for the overall performance. Some works report improvements of 1.1-2.0 Bleu $^{[\%]}$ absolute [Feng & Liu$^{+}$ 16, Tu & Lu$^{+}$ 16] while other investigations fail to show a performance increase [Peter 20]. In this work, we aim to advance the debate on whether additional dependencies to the cross-attention layer are warranted.

We reproduce existing approaches and propose new extensions to the cross-attention layer. In this work, we aim to improve a state-of-the-art machine translation model, namely the transformer architecture [Vaswani & Shazeer$^{+}$ 17]. This is in contrast to almost all prior attention extensions, which operate on a recurrent architecture. Unlike these attention-based models [Bahdanau & Cho$^{+}$ 15], the transformer uses several cross-attention layers as well as multi-head attention within each layer. It employs strictly multiplicative attention while previous models often considered additive attention layers. Furthermore, the transformer is a purely feed-forward model without any recurrent layers. This means that no information about the previous attention decision is input to the computation of the attention layer. In statistical terms, it means that the attention layer is of zero-order dependency. In contrast to this, many count-based alignment models use higher-order dependencies [Vogel & Ney$^{+}$ 96, Och & Ney 03]. Hence, we focus on the question of whether the attention mechanism benefits from additional dependencies, derived from a higher-order model.

## 1.4 Publications

The following scientific publications were all published by the author in cooperation with colleagues at peer-reviewed conferences during the time of this dissertation.

Directly related to the work in the dissertation are:

- *Recurrent Attention for the Transformer* [Rosendahl & Herold$^+$ 21]
- *Language model fusion for automatic speech recognition* [Wynands & Michel$^+$ 22]

In addition, the author made major contributions to the following publications, which, however, are not directly related to the content of this dissertation.

**Machine translation**

- *Analysis of Positional Encodings for Neural Machine Translation* [Rosendahl & Tran$^+$ 19]
- *Integrated Training for Sequence-to-Sequence Models Using Non-Autoregressive Transformer* [Tokarchuk & Rosendahl$^+$ 21]
- *Locality-Sensitive Hashing for Long Context Neural Machine Translation* [Petrick & Rosendahl$^+$ 22]

**Data filtering for machine translation**

- *The RWTH Aachen University Filtering System for the WMT 2018 Parallel Corpus Filtering Task* [Rossenbach & Rosendahl$^+$ 18]
- *Data Filtering using Cross-Lingual Word Embeddings* [Herold & Rosendahl$^+$ 21]
- *Detecting Various Types of Noise for Neural Machine Translation* [Herold & Rosendahl$^+$ 22]

**International evaluation campaigns on machine translation**

- *The RWTH Aachen University English-German and German-English Machine Translation System for WMT 2017* [Peter & Guta$^+$ 17] (minor contribution)
- *The RWTH Aachen Machine Translation Systems for IWSLT 2017* [Bahar & Rosendahl$^+$ 17]
- *The RWTH Aachen University Supervised Machine Translation Systems for WMT 2018* [Schamper & Rosendahl$^+$ 18]
- *The RWTH Aachen University Machine Translation Systems for WMT 2019* [Rosendahl & Herold$^+$ 19]

Lastly, the author participated as a co-author in the following publication:

- *Learning Bilingual Sentence Embeddings via Autoencoding and Computing Similarities with a Multilayer Perceptron* [Kim & Rosendahl$^+$ 19]

As required by §5.6 of the doctoral guidelines of the RWTH Aachen University, a description of the individual contribution of the author to his publications related to this dissertation is given in Sections 4.5 and 5.6.

# 2. Scientific Goals

In this thesis, we aim to answer a series of research questions, based on two major topics.

## Monolingual Data in Machine Translation

For common translation tasks, there is a lot more monolingual data available than bilingual data, e.g. for the Romanian→English task considered in this thesis we use around ninety times more monolingual than bilingual target data. In this work we answer the following scientific questions:

- Is monolingual data helpful to train a machine translation system?
- What is the best way to incorporate monolingual data?

These are the two major research questions that guide all investigations in Chapter 4. In answering those, a series of sub-questions arise, some of which related only to certain approaches:

- Does monolingual data allow for a better representation of the target language? Or is it only helpful to adapt to a specific domain that is not sufficiently represented in the bilingual training corpus?
- There is no established way to use monolingual source data in a machine translation system. How can we use source data and does it improve translation performance?
- Language model fusion incorporates monolingual data via an explicit language model into the translation model. How can we obtain a well-formed probability distribution from the two models? Does the normalization have an impact? Is it beneficial to train the translation and language model jointly?
- Pre-training approaches introduce a new loss which is used to optimize a subset of the model parameters. These parameters are used as an initialization for the actual main training. We investigate how to pre-train a maximum number of parameters. This includes the search for a suitable sub-task, for which a subset of the model parameters and a loss needs to be defined.
- Back-translation [Bertoldi & Federico 09, Sennrich & Haddow$^+$ 16b] is the established state-of-the-art method to include monolingual data. Can we verify the results from the literature in our work?
- For all approaches this culminates in the question: How much do they help the translation performance?

### Encoder-Decoder Cross-Attention

We also investigate the cross-attention layer that connects the encoder to the decoder and is often interpreted as a replacement for an alignment model. Many alignment models used first- or higher-order dependencies [Vogel & Ney$^+$ 96, Och & Ney 03] when modeling the relation between source and target, while the cross-attention layer obtains no explicit information from the previous target time step. There is no consensus in the scientific literature on whether cross-attention benefits from higher-order dependencies, with some works finding strong improvements [Feng & Liu$^+$ 16, Tu & Lu$^+$ 16] and others reporting no significant impact [Peter & Guta$^+$ 17, Peter 20]. We provide a new perspective by investigating the cross-attention within the transformer architecture [Vaswani & Shazeer$^+$ 17]. This is an important difference from previous works because the transformer introduced several crucial changes to all attention layers. With this in mind, we focus on the following core question throughout Chapter 5:

- Does cross-attention benefit from additional dependencies and information?

Answering this raises several related questions concerning the cross-attention layer:

- What information is beneficial?
- How should the additional dependency be represented?
- How should the additional dependency be included?
- Do the observations from recurrent translation systems [Feng & Liu$^+$ 16, Tu & Lu$^+$ 16, Peter 20] carry over to the transformer architecture?

We evaluate all our proposed methods on publicly available tasks, which allows comparison with findings from other researchers. Since the effects of adding monolingual data might depend on the environment of the language pair, we specifically focus on a variety of scenarios with respect to the available training data, ensuring that low- and high-resource tasks are considered and that the training data originates from different domains across the tasks.

# 3. Preliminaries

In this chapter, we introduce the terminology and concepts of statistical machine translation that are used throughout this work. This includes a general background of machine translation as well as an in-depth presentation of the state-of-the-art architecture.

## 3.1 Terminology and Notation

Machine translation aims to create a system that is able to automatically convert a sentence from a *source language* to the desired *target language*, producing an output sentence that is valid in the intended language while preserving the original meaning. In statistical machine translation (SMT; see Section 3.2) the translation software relies on a statistical model that is optimized or *trained* on a *bilingual training corpus* $\mathcal{T}$ that consists of sentence pairs. Each pair consists of two sequences

$$\begin{aligned} f_1^J &:= f_1, f_2, \dots, f_j, \dots, f_J && \text{source sentence} \\ e_1^I &:= e_1, e_2, \dots, e_i, \dots, e_I && \text{target sentence} \end{aligned}$$

where $f_j$ and $e_i$ are the source and target words, respectively, and the notation $f_1^J$ denotes the sequence of $J$ source words. In general, we use $f$ and $e$ to denote words of the source and target language respectively. An *n-gram* refers to a sequence of $n$ words, usually a subsequence of a sentence, e.g. $e_{i-2}^i = e_{i-2}, e_{i-1}, e_i$ describes a 3-gram selected from the target sentence $e_1^I$.

In this work we only consider bilingual corpora that are *sentence-aligned*, meaning that the source $f_1^J$ and target sentence $e_1^I$ are translations of each other. Within a sentence pair $(f_1^J, e_1^I)$ a *target-to-source word alignment* $b$ can be created by assigning each target $e_i$ word a corresponding source word $f_{b_i} = f_{b(e_i)}$. The idea behind this is that each target word is put into the target sentence because it translates some aspect of the source. An alignment expresses this by assigning each target word a source word. In this work we do not investigate explicit alignment models as they are not part of state-of-the-art architectures; however, the idea of a word alignment has been guiding the research on statistical machine translation for many years and concepts deduced from alignments are still central in the architecture and interpretation of modern machine translation systems.

Bilingual corpora typically consist of human-created translations, meaning that one sentence from each pair is created by a native speaker and then translated by a human translator. However, note that the direction of the human translation does not need to coincide with the direction of the translation system.

The progression of machine translation systems is evaluated using *development* or *evaluation sets*, sometimes shortened to *dev* or *eval set*. These terms refer to a dataset that has no or very little sentence-pair overlap with the original bilingual training corpus. The development set is used to guide design decisions during the optimization of the model, allowing the comparison of different hyperparameters, etc. In contrast to this, *test sets* are not used in the creation of the model. Instead, test sets are used to compare the performance of different systems across models and research groups. After training, the machine translation system translates each source sentence of a certain test set to generate a list of *hypothesis* translations. These are compared against the gold standard translations provided by the test set using an *evaluation measure* or *metric* (see 3.4), resulting in a performance score. The correct translation is called the *reference.*

In this work, we investigate the use of *monolingual training data* $\mathcal{T}_{\text{mono}}$, i.e. a corpus of sentences in either the source or the target language. We use $\mathcal{T}_{\text{src}}$ or $\mathcal{T}_{\text{trg}}$ to specify a monolingual training corpus of the corresponding language.

Next, we present a brief background on statistical machine translation in general and an in-depth description of state-of-the-art neural machine translation in particular.

## 3.2 Statistical Machine Translation

Statistical machine translation is a sub-task of the field of natural language processing (NLP) that uses statistical methods to model translation as a probabilistic problem. Formally translation is described as finding the target sentence $\hat{e}_1^{\hat{I}}$ that minimizes the decision errors by maximizing the *true translation probability* $\Pr(\bullet|f_1^J)$

$$f_1^J \mapsto \hat{e}_1^{\hat{I}} := \hat{e}_1^{\hat{I}}\left(f_1^J\right) := \operatorname*{arg\,max}_{e_1^I, I} \left\{ \Pr(e_1^I|f_1^J) \right\}.$$

This means we assume a true probability distribution $\Pr(e_1^I|f_1^J)$ that is conditioned on the source sentence $f_1^J$ and describes for each possible target sentence $e_1^I$ how likely it is as a translation of the source sentence $f_1^J$. A translation is obtained by applying Bayes' decision rule on the true distribution.

From this theoretical framework several challenges emerge. Notably, the true probability distribution Pr is a theoretical concept and not available in practice. Instead, we rely on statistical models $p$ that are optimized on training data to guide the decision-making process in translation. The statistical modeling is faced with three major problems [Ney 01]:

- The **modeling problem** considers the question of how a model $p$ should be formulated to approximate Pr.
- The **training problem** arises since most models $p = p_\theta$ rely on a set of free parameters $\theta$ that govern the behavior of the model. This training aims to find an assignment for each free parameter such that $p_\theta$ mimics Pr as well as possible. Since Pr is not directly observable, it is estimated from a sufficiently big set of training data.
- The **search problem** originates from the arg max operation over all possible target sequences. Performing this operation naively is infeasible in practice, since e.g. $30,000$ target words can be arranged in $5.9 \times 10^{44}$ different sequences of ten words.

The main focus of this work lies on the training and modeling problems.

In this work, we consider two types of statistical models: language models and translation models. A language model $p_{\text{LM}}$ describes the prior probability $p_{\text{LM}}(e_1^I) = p(e_1^I)$ that the sentence $e_1^I$ is a valid sentence in the language under consideration. The translation probability $p_{\text{TM}}(e_1^I|f_1^J) = p(e_1^I|f_1^J)$ models the posterior probability that the source sentence $f_1^J$ is translated to $e_1^I$. Since both language and translation models are sequential models, we can decompose the probability distribution of both by the chain rule of conditional probabilities

$$p_{\text{TM}}(e_1^I|f_1^J) := \prod_{i=1}^{I} p_{\text{TM}}(e_i|e_0^{i-1}, f_1^J)$$

where an artificial symbol $e_0 := \langle\text{BOS}\rangle$ is added to the target to mark the beginning of the sequence. A similar symbol $e_{I+1} := \langle\text{EOS}\rangle$ is appended to the end of the sequence to denote the end of the sentence. For simplicity of notation, we shift $I$ by one and consider the end-of-sequence token to be in the last position.

The presented sentence decomposition is very general and has been used to segment sentences as sequences of phrases [Zens & Och$^+$ 02], words [Brown & Pietra$^+$ 93], sub-words [Sennrich & Haddow$^+$ 16c], characters [Costa-jussà & Fonollosa 16], bytes [Costa-jussà & Escolano$^+$ 17] or a hybrid of different segmentations [Wang & Cho$^+$ 20, Carrión-Ponz & Casacuberta 22]. State-of-the-art architectures can work with most of these sentence segmentations and we use the word *token* as a general term to describe the individual units of the input and output sequence.

### 3.2.1 Neural Machine Translation

Over the last three decades, a variety of approaches were proposed for how to model the translation probability $p_{\text{TM}}$. Early works relied on word-level models with explicit alignment and language models [Brown & Pietra$^+$ 93]. Phrase-based models extended the translation model such that several consecutive words are translated as a single unit [Zens & Och$^+$ 02, Koehn & Och$^+$ 03]. Both modeling approaches are traditionally implemented as count-based models, meaning that they estimate the probability of an event by counting how often it occurred in the training data. Sentences are broken down into sequences of fixed length and their co-occurrence across the training data is counted. In order to model complex events such as long word sequences these count-based models typically require a sparse probability table, together with smoothing strategies such as backing-off [Kneser & Ney 95].

In contrast to this, neural machine translation systems operate on word sequences of arbitrary length with implicit smoothing techniques. The underlying artificial neural network extracts information from the training data and stores it within the matrices that comprise its layers. Most neural translation architectures consist of two major parts: an encoder that extracts features from the input signal, i.e. the input sentence, and a decoder that produces a series of probability distributions over the target language vocabulary. Early systems featured an architecture where the encoder and decoder were connected with a simple adapter, which caused an information bottleneck [Sutskever & Vinyals$^+$ 14]. The introduction of the cross-attention layer to connect the encoder and the decoder created a breakthrough [Bahdanau & Cho$^+$ 15, Luong & Pham$^+$ 15] and established neural machine translation systems as the new state-of-the-art method [Bojar & Chatterjee$^+$ 16]. In the next section, we focus on the current state-of-the-art self-attentive neural machine translation system called *transformer* [Vaswani & Shazeer$^+$ 17].

### 3.2.2 Building Blocks

Here, we describe the components that are the building blocks of state-of-the-art neural machine translation and language model architectures. Neural machine translation models use an encoder-decoder architecture, i.e. they consist of two major components. The task of the encoder is to map the source sentence to a sequence of hidden states

$$h_1^J := \textsc{Encoder}\left(f_1^J\right) \tag{3.1}$$

that captures all relevant information of the source sentence. The decoder obtains this representation as well as the current target context and returns a probability distribution over the target vocabulary

$$p_{\text{TM}}(e_i|f_1^J, e_0^{i-1}) = \textsc{Decoder}(h_1^J, e_0^{i-1})$$

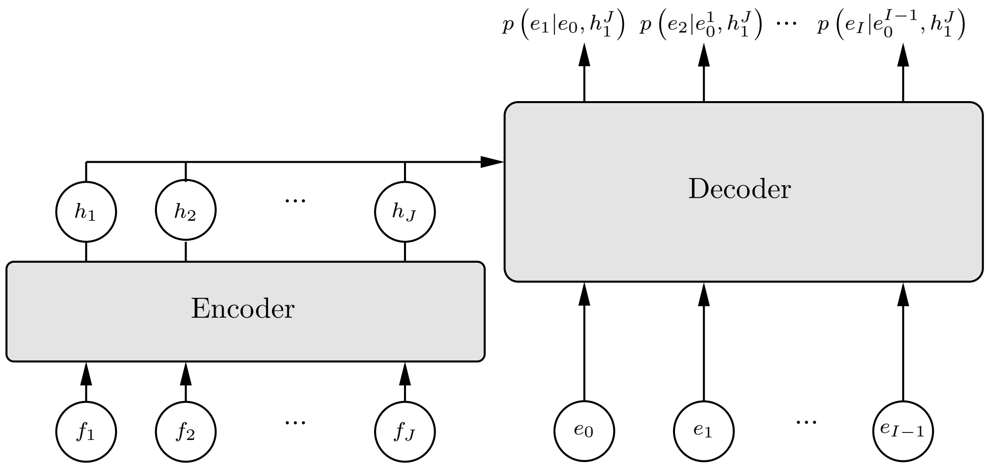

$p\left(e_1|e_0, h_1^J\right)$
$p\left(e_2|e_0^1, h_1^J\right)$
$\cdots$
$p\left(e_I|e_0^{I-1}, h_1^J\right)$
Decoder
Encoder
$h_1$
$h_2$
$h_J$
$f_1$
$f_2$
$f_J$
$e_0$
$e_1$
$e_{I-1}$

Figure 3.1: Structure of the encoder-decoder architecture. The encoder maps the input sequence $f_1^J$ to the representation $h_1^J$. The decoder generates a sequence of probability distributions based on these representations as well as the partial target sentence $e_0^{i-1}$.

which is used to generate a target sentence. This structure is visualized in Figure 3.1.

In neural machine translation, the encoder and decoder together build an artificial neural network. This neural network consists of different layers, i.e. differentiable functions, that are stacked upon each other, by the composition of their corresponding functions. The layers of modern machine translation systems are rather complex (see Section 3.2.3), consisting of sub-layers with more elemental computations. We start by introducing these general-purpose sub-layers, such as embedding, attention and feed-forward layers, before depicting the full architecture.

### Embedding Layer

The first layer of both the encoder and the decoder is a word embedding or, more precisely, a token embedding layer. Neural network-based models operate on a closed vocabulary of source $V_f$ and target tokens $V_e$. Note that this does not imply a closed vocabulary at the language level. For example, a fixed set of characters as source model-vocabulary $V_f$ can cover an unlimited number of source words. In particular, such a system can generalize to previously unseen words, meaning that the technical vocabulary of the model is closed but the vocabulary it covers in a language is open. For further discussion on these approaches we refer the reader to Section 3.3.2. In this work, we use the term vocabulary to refer to the technical vocabulary of the model. The entries of the vocabulary are the tokens of the segmented sentences.

No matter whether the tokens of the vocabulary are characters, sub-words or words, they are always discrete symbols, while neural networks operate fundamentally in a continuous space of hidden states. An embedding maps these discrete input tokens to continuous representations which can be trained with the rest of the model (see Section 3.2.4). This is achieved by representing the source token $f \in V_f$ from the source vocabulary as a *one-hot vector*

$$\mathbb{1}_f := \begin{cases} 1 & v = f \\ 0 & \text{else} \end{cases} \qquad \forall\, v \in V_f.$$

The resulting vector $\mathbb{1}_f \in \mathbb{R}^{|V_f|}$ is multiplied with an embedding matrix $E_f \in \mathbb{R}^{d_{\text{model}} \times |V_f|}$

$$\tilde{f} := E_f \mathbb{1}_f \tag{3.2}$$

i.e. it is passed through a fully connected linear layer without an activation function. Note that this multiplication is functionally equivalent to selecting the $f$-th column of the matrix $E_f$.

### Positional Encoding

Given a source sentence $f_1^J$, each word embedding $\tilde{\tilde{f}}_j$ is a context-free representation of the corresponding source token $f_j$. In recurrent neural networks [Sutskever & Vinyals$^+$ 14], the sequence order of the input sentence is implicitly encoded by the order of operations. In contrast to this, all sequence layers in self-attentive encoders are invariant to the order of their input sequence. A permutation of the input sequence simply causes an identical permutation of the output sequence. Since the order of words in a sentence does matter, the transformer [Vaswani & Shazeer$^+$ 17] relies on an explicit positional encoding

$$\tilde{f}_j := \tilde{\tilde{f}}_j + \mathrm{pos}(j) \tag{3.3}$$

where the $j$-th position of the input sequence is encoded in the positional vector

$$\mathrm{pos}(j)_k := \begin{cases} \sin\left(\frac{j}{10000^{2k/d_{\mathrm{model}}}}\right) & k \text{ is even} \\ \cos\left(\frac{j}{10000^{(2k-1)/d_{\mathrm{model}}}}\right) & k \text{ is odd} \end{cases} \qquad \forall k \in \mathbb{R}^{d_{\mathrm{model}}}.$$

These sinusoidal positional embeddings are not trainable. Instead, they rely on fixed sine and cosine functions with different frequencies. Since the encoding of a certain position $j$ is not a learned vector, it is possible to generate encodings of positions that are not observed during training. Hence the sinusoidal positional embeddings are capable of generalizing to sequences of arbitrary length.

We sometimes include layer normalization [Ba & Kiros$^+$ 16] in the positional encoding

$$\tilde{f}_j := \mathrm{LayerNorm}(\tilde{f}_j)$$

which can make the training more stable. We denote explicitly if layer normalization is applied.

#### Attention

The attention mechanism is the core component that fueled two of the last major jumps in machine translation performance [Bahdanau & Cho$^+$ 15, Vaswani & Shazeer$^+$ 17].

Attention is often interpreted as a mapping that performs a soft lookup of a query vector on a sequence of key, value pairs. The input is a sequence of *query vectors*

$$q_1, \ldots, q_I \in \mathbb{R}^{d_q}$$

together with a sequence of *key* and *value* pairs

$$(k_1, v_1), \ldots, (k_J, v_J) \in \mathbb{R}^{d_k} \times \mathbb{R}^{d_v}.$$

In the case of a 'hard' lookup table, the value $v_j$ is returned if its key $k_j$ matches the query $q_i$ exactly

$$\mathrm{lookup}(q_i; k_1^J, v_1^J) := \begin{cases} v_j & \exists j : k_j = q_i \\ \langle\text{not found}\rangle & \text{else.} \end{cases}$$

Instead, attention performs a soft lookup where the current query $q_i$ is compared against each key $k_j$ via a similarity measure $\hat{\alpha}$

$$\hat{\alpha}_{i,j} := \hat{\alpha}(q_i, k_j) := \frac{1}{\sqrt{d}}(W_k k_j)^{\mathsf{T}} W_q q_i \tag{3.4}$$

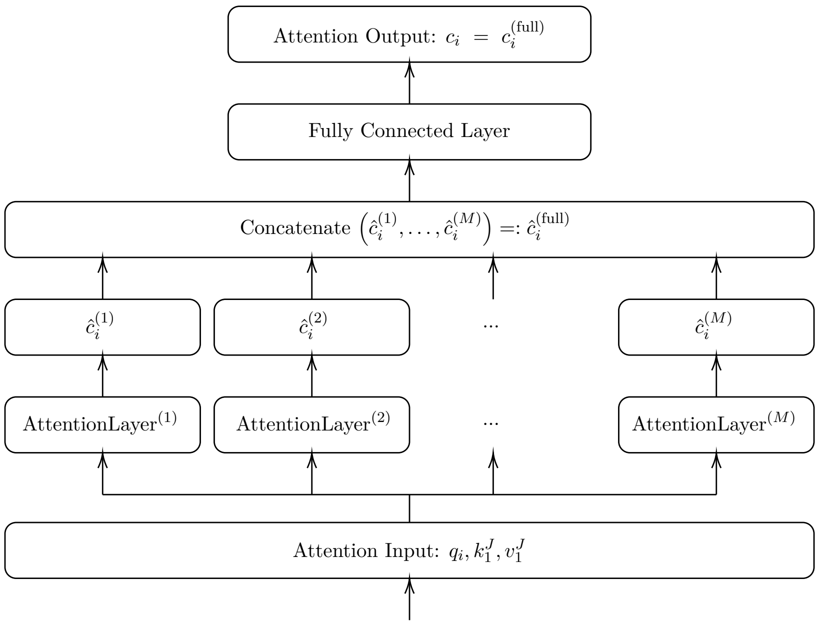

Attention Output: $c_i = c_i^{(\text{full})}$
Fully Connected Layer
Concatenate $\left(\hat{c}_i^{(1)}, \dots, \hat{c}_i^{(M)}\right) =: \hat{c}_i^{(\text{full})}$
$\hat{c}_i^{(1)}$
$\hat{c}_i^{(2)}$
...
$\hat{c}_i^{(M)}$
AttentionLayer$^{(1)}$
AttentionLayer$^{(2)}$
...
AttentionLayer$^{(M)}$
Attention Input: $q_i, k_1^J, v_1^J$

Figure 3.2: Multi-head attention layer with $M$ heads.

with weight matrices $W_q \in \mathbb{R}^{d \times d_q}$ and $W_k \in \mathbb{R}^{d \times d_k}$. These *attention energies* $\hat{\alpha}$ assign a measure of importance and similarity to each key. Normalizing them yields the *attention weights*

$$\alpha(j|i) := \frac{\exp\left(\hat{\alpha}_{i,j}\right)}{\sum_{j'} \exp\left(\hat{\alpha}_{i,j'}\right)}, \tag{3.5}$$

a probability distribution over the positions $J$. Each value vector $v_j$ should influence the output of the soft lookup proportionally to its similarity to the query vector $q_i$. Hence, the attention outputs a weighted average over all values

$$c_i := \sum_j \alpha(j|i) W_v v_j$$

Overall, the attention layer is a function

$$c_i = \text{AttentionLayer}\left(q_i, k_1^J, v_1^J; W_q, W_k, W_v\right).$$

This general attention layer is used in various ways in the transformer architecture.

## Multi-Head Attention

The transformer architecture introduces multi-head attention [Vaswani & Shazeer⁺ 17]. Instead of using a single attention layer, a total of $M$ attention layers operates on the same input in parallel. These individual layers are commonly called *attention heads* and each uses its own set of weight matrices $W_q^{(m)}, W_k^{(m)}, W_v^{(m)}$

$$\begin{aligned} c_i^{(m)} &:= \text{AttentionLayer}^{(m)}\left(q_i, k_1^J, v_1^J\right) \\ &:= \text{AttentionLayer}\left(q_i, k_1^J, v_1^J; W_q^{(m)}, W_k^{(m)}, W_v^{(m)}\right) \end{aligned}$$

The output of all $M$ attention heads is concatenated to a new hidden state

$$\hat{c}_i^{(\text{full})} := \begin{pmatrix} c_i^{(1)} \\ \vdots \\ c_i^{(M)} \end{pmatrix}$$

which is passed through a fully connected layer

$$c_i^{(\text{full})} := W_{\text{fc}} \hat{c}_i^{(\text{full})}$$

without an activation function or a bias term. Figure 3.2 illustrates the multi-head attention sub-layer. Since multi-head attention showed solid improvements [Vaswani & Shazeer$^+$ 17], all systems in this work use multi-head attention for all attention components.

### Self-Attention

Self-attention layers are a special case of general attention where key, query and value are identical

$$q_1^I = k_1^I = v_1^I.$$

However, their corresponding weight matrices are still distinct. Self-attention layers are sequential feed-forward layers; this distinguishes them from the recurrent sequence layers such as LSTMs [Hochreiter & Schmidhuber 97] and GRUs [Cho & van Merrienboer$^+$ 14] previously used in neural machine translation. Since there is no dependency between the computation of $c_i^{(\text{full})}$ and $c_{i+1}^{(\text{full})}$, these values can be calculated in parallel.

Self-attention layers that access the full key sequence for each query are called bidirectional. In many text-generation tasks, however, unidirectional self-attention layers are required. These layers restrict the query $q_i$ to only access the previous $i' \leq i$ key, value pairs $(k_{i'}, v_{i'})$. This can be achieved by modifying the attention energies from Equation 3.4 to assign artificially low values to all other positions

$$\hat{\alpha}_{i,i'} = \begin{cases} \frac{1}{\sqrt{d}} (W_{i'} k_{i'})^\intercal W_q q_i & i' \leq i \\ -\infty & \text{else.} \end{cases}$$

This means that after the normalization described in Equation 3.5 all future positions $i' > i$ obtain an attention weight of zero. This approach is also sometimes called *causal attention* or *left-to-right attention.*

### Feed-Forward Block

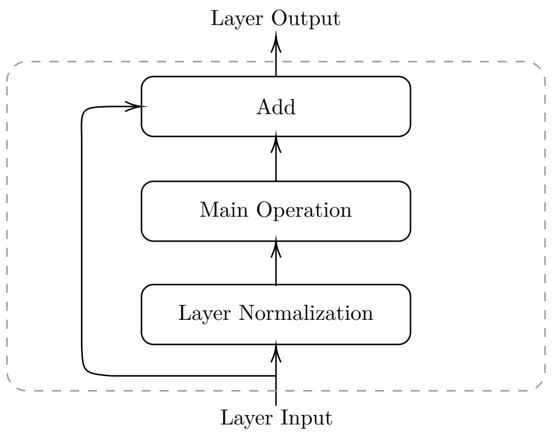
The transformer architecture features advanced linear layers that consist of two stacked, fully connected layers. The first fully connected layer, with a rectified linear unit (ReLU) as activation function and parameters $W_1, b$, projects the input vector $x \in \mathbb{R}^{d_{\text{model}}}$ to a higher dimension

$$\begin{aligned} \hat{x} &:= \text{FullyConnected}(x) \\ &= \text{ReLU}\left(W_1 x + b\right) \in \mathbb{R}^{d_{\text{ff}}} \end{aligned}$$

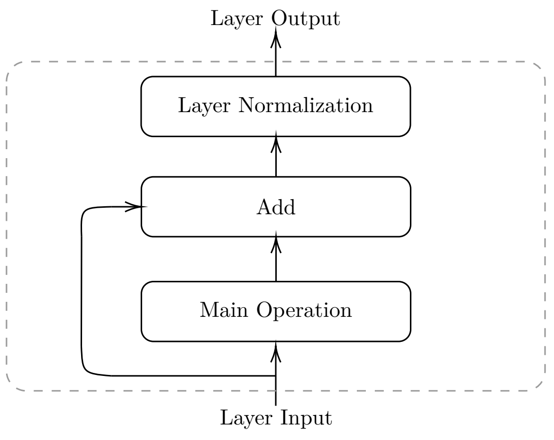
where commonly $d_{\text{ff}} = 4 \times d_{\text{model}}$. Next the intermediate state $\hat{x}$ is mapped back to its input dimension $d_{\text{model}}$ via a fully connected layer without activation function and bias vector

$$\begin{aligned} y &:= W_2 \hat{x} \\ &= W_2 \cdot \text{ReLU}\left(W_1 x + b\right) \end{aligned}$$

where $W_2$ denotes the weight matrix.



(a) Pre-Norm



(b) Post-Norm

Figure 3.3: Structure of most transformer sub-layers, namely self-attention, cross-attention and feed-forward blocks. Embedding, projection and positional encoding layers are used without modifications. In contrast with the original transformer architecture [Vaswani & Shazeer$^+$ 17] that utilizes the post-norm approach (b), we use the pre-norm (a) for all experiments.

### 3.2.3 Transformer

In this section, we describe the transformer architecture as introduced in 2017 by Vaswani et al. [Vaswani & Shazeer$^+$ 17]. This architecture is so universally accepted as state-of-the-art, that in 2021 all 170 submitted solutions to the prestigious shared task on machine translation of the Conference on Machine Translation (WMT) used a variant of it [Akhbardeh & Arkhangorodsky$^+$ 21].

The transformer consists of two major parts, the encoder and the decoder, each of which is composed of several sub-layers. In the following, we describe the structure of these sub-layers before introducing the architecture of the encoder and decoder.

#### Structure of a Sub-Layer

Self-attention, cross-attention and feed-forward blocks are essential components of the transformer architecture. Each of these is wrapped in the sub-layer structure depicted in Figure 3.3(a). Layer normalization [Ba & Kiros$^+$ 16] is applied on the input $x_1^I$ of the layer

$$\hat{x}_1^I = \text{LayerNorm}(x_1^I).$$

These layer-normalized hidden states are processed by the main operation of the layer

$$\hat{\hat{x}}_1^I = \text{MainOperation}(\hat{x}_1^I)$$

where the main operation is either an attention layer or a feed-forward block. The original input $x_1^I$ of the sub-layer is added to the hidden state $\hat{\hat{x}}_1^I$

$$y_i = \hat{\hat{x}}_i + x_i \qquad \forall 1 \leq i \leq I$$

forming a *residual connection* or *skip connection* that facilitates the training of deep neural architectures [He & Zhang$^+$ 16]. The sub-layer outputs the sequence $y_1^I$. Due to the residual connection, each vector $y_i$ in the output sequence has the same dimension as its corresponding

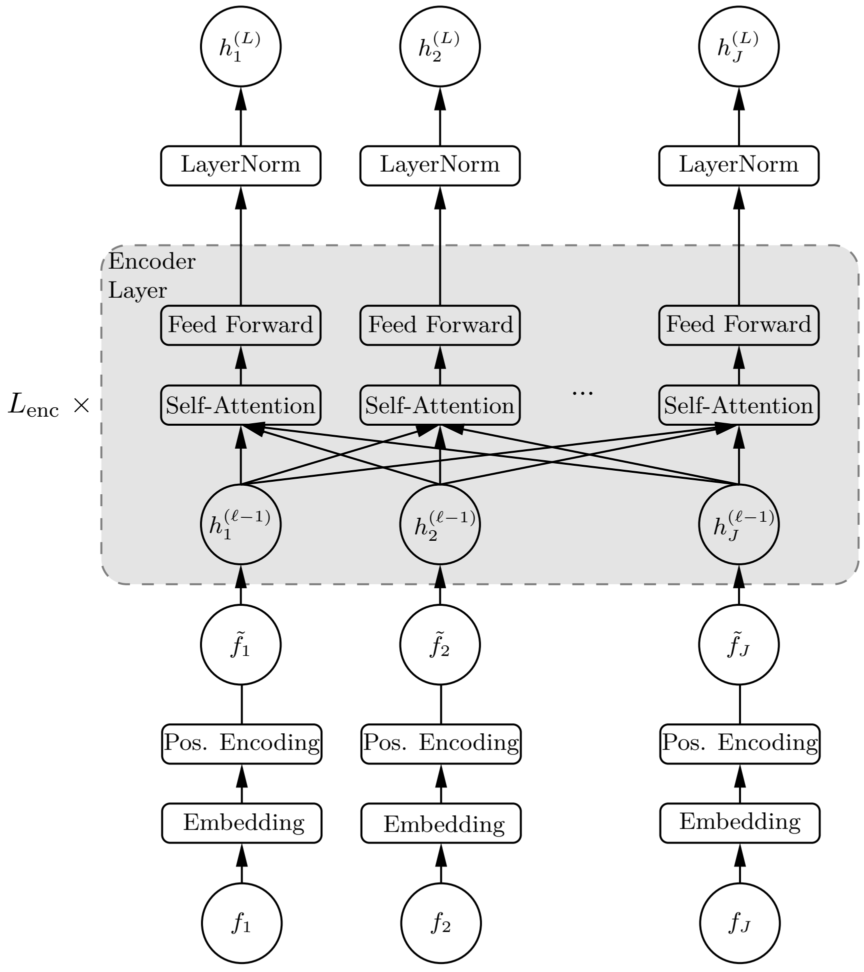


Figure 3.4: Encoder of the transformer architecture with $L_{\text{enc}}$ layers. Sub-layers with computational operations are denoted in rectangles and important hidden states are shown in circles. The self-attention layer and feed-forward block use the structure of linear normalization and residual connection with the pre-norm approach as described in Figure 3.3(a).

input vector $x_i$. Hence, the hidden state between two transformer sub-layers is always of the same dimension $d_{\text{model}}$.

Note that the *pre-norm* sub-layer structure as described above differs from the *post-norm* approach used in the original transformer paper, where layer normalization is instead applied after the addition operation, as depicted in Figure 3.3(b). Different works investigate the connection between layer normalization, training stability and the use of warm-up steps in training [Xiong & Yang$^+$ 20, Huang & Pérez$^+$ 20, Nguyen & Salazar 19]. Since pre-norm architectures are found to be more stable in training and do not require warm-up steps, they are used in all our experiments.

### Encoder

The transformer encoder consists of an embedding, a positional encoding and a stack of encoder layers as depicted in Figure 3.4. The input sequence $f_1^J$ is first passed through an embedding layer before adding a positional encoding. The resulting sequence of embedded tokens $\tilde{f}_1^J$ is passed on to the stack of encoder layers.

Conventionally a neural network layer contains a single, relatively simple operation. However, in the context of the transformer architecture, the term 'encoder layer' refers to a rather complex structure, consisting of several elaborate sub-layers. Each encoder layer consists of a bidirectional self-attention layer followed by a feed-forward block layer, as shown in Figure 3.4. Both of these operations are individually wrapped in a sub-layer block using layer normalization and a residual connection with the pre-norm approach (see Figure 3.3(a)). The self-attention layers are multi-headed with $M$ heads and the number of encoder layers stacked is denoted by $L = L_{\text{enc}}$ where no parameters are shared between the layers. We denote the input of layer $\ell \in \{1, \ldots, L_{\text{enc}}\}$ by $(h_1^J)^{(\ell-1)} = h^{(\ell-1)}$ and initialize

$$h^{(0)} = \left(h^{(0)}\right)_1^J = \tilde{f}_1^J$$

using the sequence of the embedded source tokens $\tilde{f}_1^J$. Since we use a pre-norm approach the output $h^{(L)}$ of the last layer is not layer-normalized. Hence, we apply a final layer normalization on top of the encoder. For simplicity, we do not display this operation in further figures and merge the positional encoding into the embedding layer.

### Decoder

The decoder closely resembles the architecture of the encoder. As shown in Figure 3.5, the target sentence $e_0^I$ is passed through an embedding layer and combined with a positional encoding before entering the first decoder layer. The term *decoder layer* refers to a structure of several complex sub-layers, the first of which is a multi-headed self-attention layer. Due to the requirements on the decoder in search, this self-attention layer is unidirectional and outputs the *decoder hidden state* $s_i = s_i^{(\ell)}$. Note that there is an index shift between the input word $e_{i-1}$ and the generated hidden state vector $s_i$. This hidden state is passed to a cross-attention layer.

**Cross-attention** [Bahdanau & Cho+ 15] was a breakthrough in machine translation systems, removing the bottleneck between the encoder and the decoder of previous architectures [Sutskever & Vinyals+ 14] and enabling neural network standalone systems to establish themselves as the new state-of-the-art model in machine translation. The cross-attention layer is a multi-head attention layer with shared keys and values

$$q_i^{(\ell)} := s_i^{(\ell)}$$
$$\left(k^{(\ell)}\right)_1^J := \left(v^{(\ell)}\right)_1^J := h_1^J$$

i.e. the queries are the decoder states and the keys and values are the encoder outputs. As a consequence, the queries are layer-specific, while the key-value pairs are shared across all layers. Cross-attention is a major focus of this work and we refer the reader to Section 5.1 for an in-depth discussion.

The output of the cross-attention layer, typically called the *context vector* $c_i = c_i^{(\ell)}$, contains information about the source and the partial target sentence. It is passed through a feed-forward block, which is the final building block of a decoder layer. All three sub-layers, namely self-attention, cross-attention and the feed-forward block, are wrapped in the pre-norm sub-structure shown in Figure 3.3(a), including a residual connection and layer normalization. The residual connection in particular means that the output of a decoder layer has the same dimensions and sequence length as its input.

In the decoder $L_{\text{dec}}$ layers are stacked upon each other and commonly we use $L_{\text{dec}} = L_{\text{enc}} = L$. After the last decoder layer, a final layer normalization is applied before a linear layer projects the hidden states to the size of the vocabulary. This *projection layer* does not use an activation function.

The output of the projection layer is normalized via a softmax operation and the result is a probability distribution over the target vocabulary $V_e$.

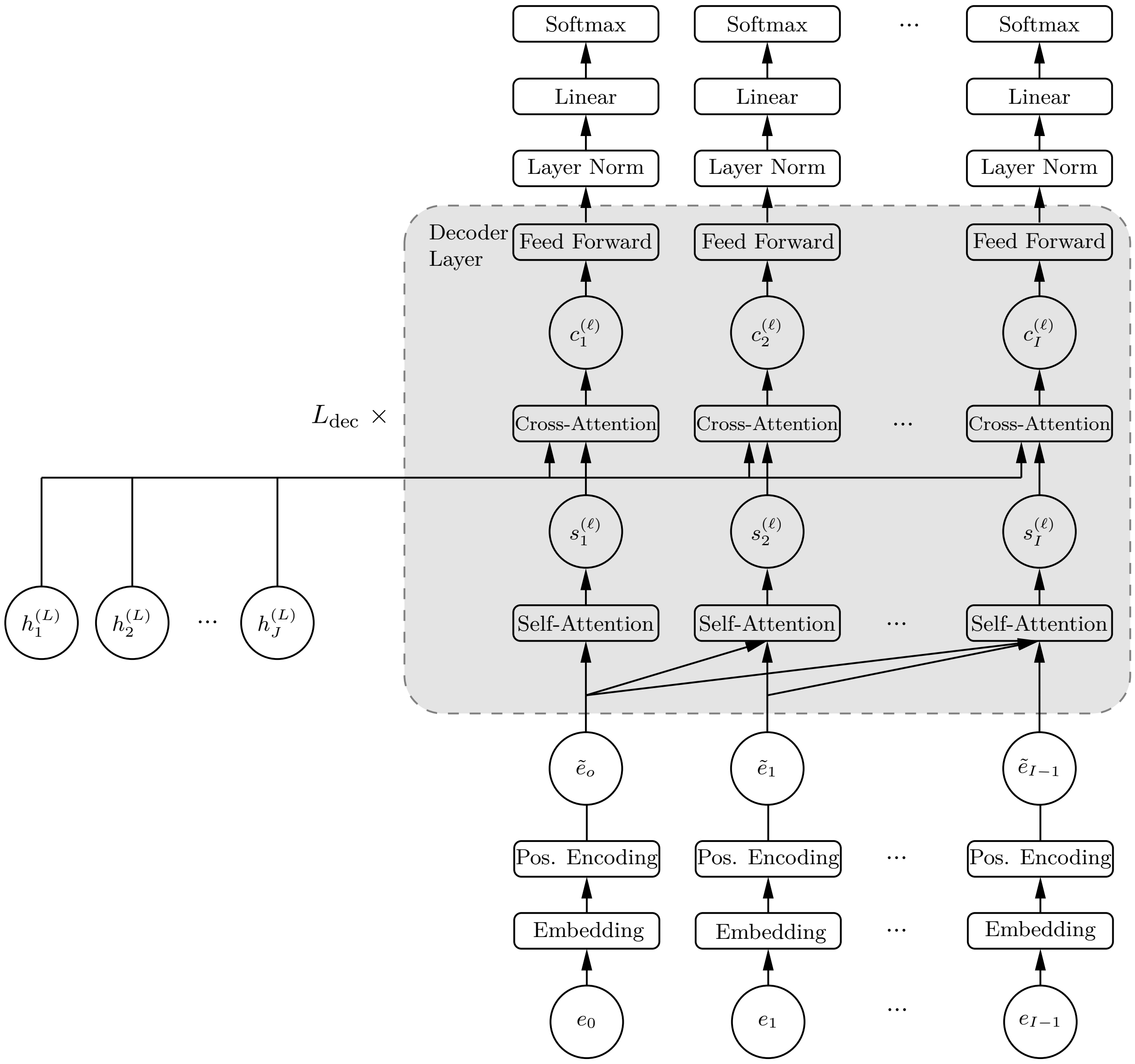


Figure 3.5: Decoder of the transformer architecture. Layers with computational operations are denoted in rectangles and important hidden states are shown in circles.

### Transformer Architecture for Translation Models

By connecting the encoder and the decoder we obtain the transformer architecture [Vaswani & Shazeer$^+$ 17] as depicted in Figure 3.6. We simplify the figure by omitting technical details such as the final layer normalization operations for the encoder and decoder. For all further representations, we merge the positional encoding into the embedding layer.

Note that the transformer is a fully feed-forward architecture, meaning that the hidden states of a layer can be computed in parallel across the whole input sequence. This yields a huge improvement in training speed and is one important reason why the transformer architecture outperforms recurrent LSTM-based systems [Zeyer & Bahar$^+$ 19].

In the transformer architecture many hyperparameters are shared between layers. For example, all hidden states output from a sub-layer are of size $d_{\text{model}}$ for each time step both in the encoder and the decoder. All self- and cross-attention layers in the transformer architecture are multi-head

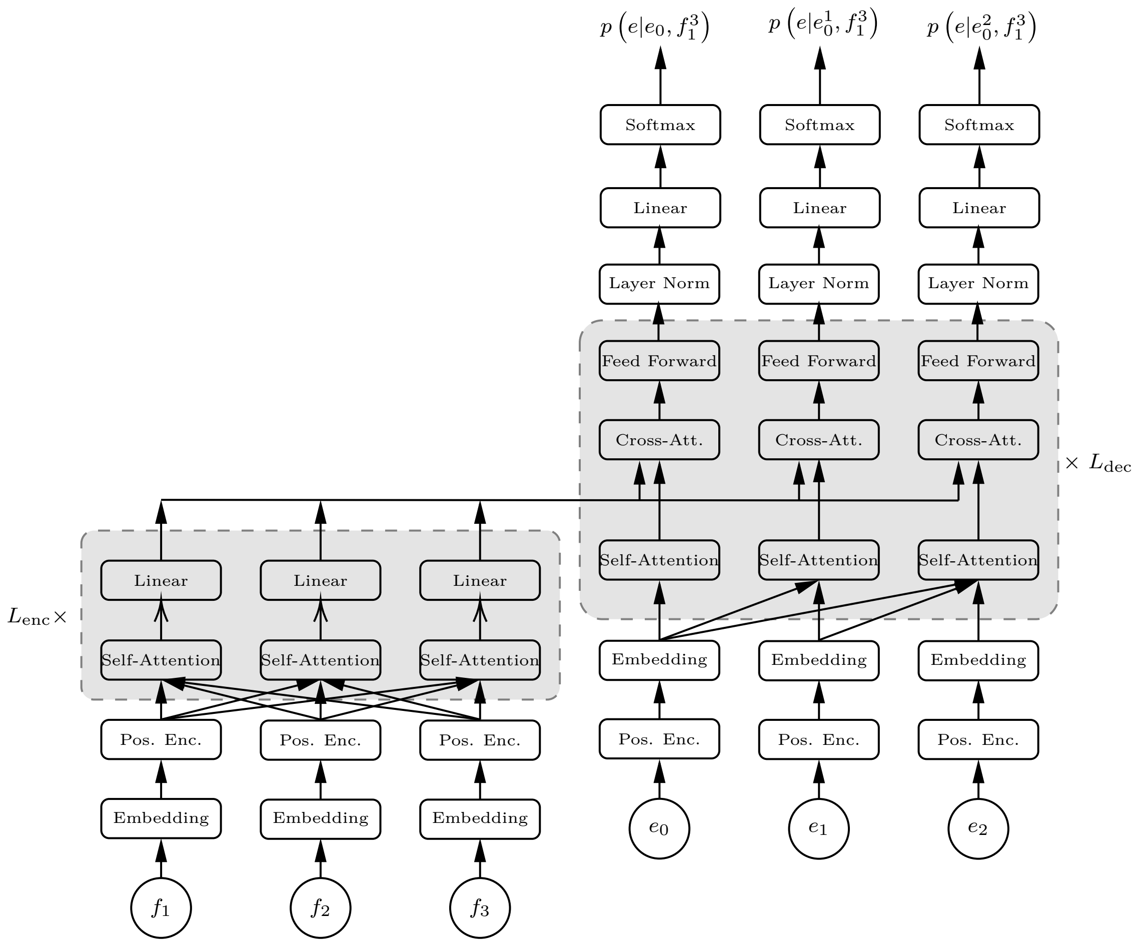


Figure 3.6: Unrolled transformer architecture for a source and target sentence of length $J = I = 3$.

with the number of heads $M$ shared across all layers and the dimension of the projected query, key or value vector is $d_{\text{head}} = \frac{d_{\text{model}}}{M}$. The feed-forward blocks use an internal dimension of $d_{\text{ff}}$ commonly set to $d_{\text{ff}} = 4d_{\text{model}}$.

### 3.2.4 Training

Most neural network layers contain free parameters, commonly in the form of weight matrices. The process of estimating the free parameters $\theta$ of a neural network is called *training*. This is commonly done by selecting a training set $\mathcal{T}$ of example data points and choosing a *training criterion* $F$ or objective function. During training an optimization algorithm iteratively finds an assignment $\theta_\tau$ of the network parameters for each *training step* $\tau$ such that the desired objective function is reduced. In the case of machine translation, each data point is a pair $(f_1^J, e_1^I)$ of source and target sentence which are translations of each other. In this work we only consider supervised training, meaning that for each sample the correct output is known during training.

In the following, we introduce the training criterion and optimizer used throughout this work.

### Training Criterion

In accordance with the literature [Sutskever & Vinyals$^+$ 14, Bahdanau & Cho$^+$ 15, Vaswani & Shazeer$^+$ 17], we use the training criterion

$$F(\theta) := \frac{1}{|\mathcal{T}|} \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log p(e_1^I | f_1^J, \theta)$$

to estimate the parameters $\theta$ of the model $p = p_\theta = p(\bullet|\theta)$ over the training data $\mathcal{T}$. Since the training criterion is only used to find the maximizing argument

$$\theta^* := \arg\max_\theta F(\theta)$$

we commonly drop the scaling factor and use the equivalent training criterion

$$F(\theta) = \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log p(e_1^I | f_1^J, \theta). \tag{3.6}$$

Maximizing $F$ is equivalent to minimizing the negative cross-entropy loss

$$\mathcal{L}_{\text{CE}}(\theta) = -\frac{1}{|\mathcal{T}|} \sum_{(f_1^J, e_1^I) \in \mathcal{T}} p_{\text{data}}(e_1^I | f_1^J) \log p(e_1^I | f_1^J, \theta) \tag{3.7}$$

where $p_{\text{data}}$ is the empirical probability of observing the translation from $f_1^J$ to $e_1^I$ on the training data. Since the empirical probability distribution focuses all probability mass on a single point for each training pair $(f_1^J, e_1^I)$, Equation 3.7 can be simplified

$$\mathcal{L}_{\text{CE}}(\theta) = -\frac{1}{|\mathcal{T}|} \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log p(e_1^I | f_1^J, \theta) \tag{3.8}$$

and the scaling factor $\frac{1}{|\mathcal{T}|}$ is often omitted since it does not affect the optimization problem.

Label smoothing [Szegedy & Vanhoucke$^+$ 16] proved to be an effective technique to reduce overfitting, i.e. learning specifics of the training data instead of the general patterns of the task. In the vanilla training criterion, we assume that the probability mass $p(\bullet|f_1^J)$ should be focused entirely on the ground truth translation $e_1^I$

$$\begin{aligned} F(\theta) &= \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log p(e_1^I | f_1^J, \theta) \\ &= \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \sum_{i=1}^{I} \sum_{e \in V_e} \delta(e, e_i) \log p(e | e_0^{i-1}, f_1^J, \theta) \end{aligned}$$

where the Kronecker delta $\delta(e, e_i)$ yields an empirical, one-hot probability distribution over the target vocabulary $V_e$. The idea of label smoothing is to take an amount of $\varepsilon > 0$ of the ground truth label and spread this probability mass uniformly across all classes $V_e$

$$F_{\text{smoothed}}(\theta) = \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \sum_{i=1}^{I} \sum_{e \in V_e} \left[ (1-\varepsilon)\delta(e, e_i) + \frac{\varepsilon}{|V_e|} \right] \log p(e | e_0^{i-1}, f_1^J, \theta). \tag{3.9}$$

Many variations of label smoothing exist [Gao & Wang$^+$ 20]; however, throughout this work, we use $F_{\text{smoothed}}$ as the training criterion if not stated differently.

### Optimizer

Optimizing the parameters of the transformer architecture requires the solution of a non-convex optimization problem with typically 50M-300M free parameters (see e.g. [Vaswani & Shazeer+ 17, Ott & Edunov+ 18, Fan & Gong+ 21]). Since no close-form solution is known for this problem, iterative methods such as stochastic gradient descent (SGD) are used to train neural models. Gradient descent methods update the parameters $\theta$ of the objective function $F(\theta)$ based on its current gradient

$$\theta_{\tau+1} := \theta_\tau - \lambda \frac{\partial F}{\partial \theta}(\theta_\tau) \tag{3.10}$$

where the *step size* or *learning rate* $\lambda$ is a hyperparameter of the training.

Since the training criterion $F$ and its gradient depend on a sum over the entire training data, an exact computation of either is infeasible. Hence, in practice, stochastic gradient methods partition the training data into *mini batches* of a fixed *batch size* and compute an approximation of the true gradient based on each sub-selection of the training data. Note that we distinguish between the technical batch size, i.e. the batch size that is computed in parallel on the hardware, and the effective batch size, which can be increased by accumulating the gradients computed from several technical batches before applying one update step as described in Equation 3.10.

The gradient of a neural network with hundreds of millions of parameters can be computed efficiently due to the layered structure of the underlying function. Stacking layers in a neural network is equivalent to the composition of differentiable functions, which allows for efficient computation of the gradient via the chain rule of calculus. This property is the core of the backpropagation algorithm [Rumelhart & Hinton+ 86] that allows computation of the gradient of big neural networks. In this work, we rely on the automatic computation of the gradient of all neural networks via backpropagation provided by the Tensorflow toolkit [Abadi & Agarwal+ 15].

The approach of parameter estimation via stochastic gradient descent with gradients computed from backpropagation is the gold standard in neural-network-based machine learning approaches. However, there are many variations to the stochastic gradient descent algorithm used in the training of neural networks in general [Sun & Cao+ 20] and machine translation in particular [Bahar & Alkhouli+ 17]. In this work, we rely on the Adam [Kingma & Ba 15] optimizer to train all machine translation models. An important parameter of the Adam algorithm is the learning rate $\lambda$. We start with a given $\lambda$ and scale it down whenever the model fails to increase validation set performance for several consecutive updates. This procedure is commonly called *newbob learning rate reduction.*

The transformer architecture is commonly trained using a series of warm-up steps during which the learning rate is increased [Vaswani & Shazeer+ 17]. After a fixed amount of warm-up steps, the learning rate can only be kept or decreased for the rest of the training. Since we use a pre-norm structure for the transformer sub-layer (see Section 3.2.3 for details), warm-up steps are not needed to obtain stable training [Nguyen & Salazar 19]. Hence, we do not use any warm-up steps, in contrast to many common implementations of the transformer architecture.

A training run with a gradient descent algorithm creates a series of model iterations, most of which are immediately overwritten by their successor state. At regular intervals, based on the number of training steps, we store the intermediate version of the model as a *model checkpoint* for evaluation. These checkpoints are evaluated by the target metric on the development set, namely Bleu (see Section 3.4.1) for machine translation models and perplexity (see Section 3.4.3) for language models, and the best checkpoint according to the respective metric is selected for further use. This procedure is equivalent to validation-based early stopping.

### 3.2.5 Search

From the training we obtain a translation model $p = p(\bullet|\theta)$. To translate a source sentence $f_1^J$ we ideally apply the Bayes' decision rule with length normalization

$$\hat{e}_1^{\hat{I}} := \underset{I, e_1^I}{\arg\max} \left\{ \sqrt[I]{p(e_1^I|f_1^J)} \right\} \tag{3.11a}$$

$$= \underset{I, e_1^I}{\arg\max} \left\{ \frac{1}{I} \log p(e_1^I|f_1^J) \right\} \tag{3.11b}$$

and select the hypothesis $\hat{e}_1^{\hat{I}}$ with the highest probability according to the translation model. However, the set of possible candidates $e_1^I$ of length $I$ is $|V_e|^I$. A small vocabulary contains around $|V_e| = 3,000$ entries and the average target sentence length is around $I_{\text{avg}} = 20$ tokens for many tasks. This means that there are $3,000^{20} = 10^{69}$ possible sequences to consider, which is infeasible in practice. Instead, we apply beam search to approximate the maximization.

Beam search is the dominant algorithm for performing translation with neural translation models [Sutskever & Vinyals$^+$ 14, Bahdanau & Cho$^+$ 15, Vaswani & Shazeer$^+$ 17]. Performing left-to-right decoding, the search algorithm starts with an empty hypothesis $e_0 = \langle BOS \rangle$. Each active hypothesis $e_0^i$ is iteratively expanded by generating the probability distribution $p(e|e_0^i)$ over the target vocabulary. The $n$ expanded hypotheses with the highest probability form the new set of active hypotheses. The parameter $n$ is the *beam size* and for $n = \infty$ beam search is an exact search algorithm. The beam search algorithm outputs an *$n$-best list* $\mathcal{B}_n$ consisting of the $n$ best hypotheses with respect to the model $p$. The sentence with the highest probability $\hat{e}_1^{\hat{I}}$ is used as the translation or *hypothesis* of the model.

### 3.2.6 Transformer Architecture for Language Models

A variant of the transformer architecture has become a very successful language model (LM) [Al-Rfou & Choe$^+$ 19, Dai & Yang$^+$ 19, Irie & Zeyer$^+$ 19]. A language model assigns a probability distribution over a vocabulary $V_e$ for a sequence of words $e_1^I$

$$p_{\text{LM}}(e_1^I) = \prod_{i=1}^{I} p_{\text{LM}}(e_i|e_0^{i-1})$$

also called the *prior*. For simplicity, we commonly denote $p = p_{\text{LM}}$. Language models measure the fluency of the sentence $e_1^I$ or, more formally, how probable it is to observe the sentence $e_1^I$ in the respective language. Similar to machine translation, the task of language modeling relies on statistical approaches, with neural networks providing state-of-the-art results. Notably, these models are trained using monolingual data, which by nature is easier to obtain than bilingual data. We refer to Appendix A.1 for a comparison between the amount of monolingual and bilingual data available across several tasks.

In this work, we only use transformer language models as described by Irie et al. [Irie & Zeyer$^+$ 19]. The architecture is depicted in Figure 3.7 and is similar to a transformer decoder without cross-attention layers. The input sequence is first passed through an embedding layer and combined with a sinusoidal positional embedding. This is followed by a stack of $L$ transformer language model layers each consisting of a unidirectional self-attention sub-layer as well as a feed-forward block. Both of these are nested in the sub-structure of transformer sub-layers as described in Section 3.2.3. In contrast to translation models, we use a post-norm architecture for all standalone language models [Irie & Zeyer$^+$ 19]. The output of the $L$-th transformer language model layer is passed through a linear layer without an activation function which projects

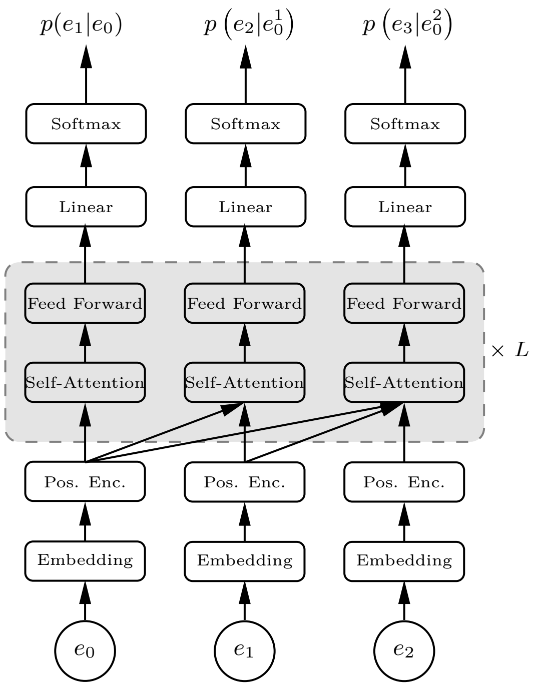


Figure 3.7: Language model based on the transformer architecture.

the hidden state to match the size of the vocabulary $V_e$. A softmax layer transforms it into a probability distribution over $V_e$.

In this work, we use language models for two main purposes: (a) as a standalone model used in log-linear model combination and (b) during multi-task training as a sub-model of a transformer translation model. Based on those two applications, two different language model setups are utilized.

For the standalone models (a) we follow the best practices for the architecture laid out by Irie et al. [Irie & Zeyer$^+$ 19]. In particular, the sub-layers use a post-norm approach and training is performed using plain stochastic gradient descent training. Furthermore, queries, keys and values share the same dimensionality across all heads and layers. However, we use an explicit positional encoding, and due to the necessity to fit the language and translation models together in the memory of a single graphics processing unit (GPU), we restrict the model size. For details regarding the model setup and size we refer the reader to Section 4.4.1

Language models used in multi-task and pre-training approaches (b) need to be compatible with the translation models, namely, they are required to be a sub-model of the transformer architecture. Hence we use pre-norm approaches for all language models involved and train them using the Adam optimizer [Kingma & Ba 15].

## 3.3 Pre-Processing

To simplify and regularize the input text we apply several pre-processing steps before passing it to the statistical models. Pre-processing is applied to the source and target sentences to reduce the system vocabularies and decrease data sparsity.

### 3.3.1 Tokenization

Tokenization refers to the act of segmenting an input sentence into a series of words and punctuation marks. In many languages, punctuation marks are directly attached to a neighboring word, e.g. a full stop is appended to the last word in the sentence while an opening quotation mark is attached to the beginning of the next word. Tokenization tools commonly add a whitespace to separate a punctuation mark from the word it is attached to. In order to distinguish these occurrences from punctuation marks that can be considered part of a token or expression, such as *e.g.* or *don't*, tokenization tools are used. Widely used tokenizers, such as the one provided by the Moses toolkit [Koehn & Hoang$^{+}$ 07], are rule-based systems that cover common patterns.

### 3.3.2 Sub-Word Units

Translation models in general and neural machine translation models in particular commonly operate on fixed (technical) vocabulary for the source and target language. In the transformer architecture this is caused by the embedding layer in which a one-hot representation $\mathbb{1}_e \in V_e$ of the target word $e$ is multiplied with the embedding matrix $E \in \mathbb{R}^{d_{\text{model}} \times |V_e|}$. Since the vocabulary $V_e$ is generated from the available training data, any word that is unknown at the time of generation of the vocabulary $\hat{e} \notin V_e$ cannot be represented by the embedding layer. Besides causing *out-of-vocabulary* (OOV) or *unknown words*, a fixed-size vocabulary frequently suffers from data sparsity. Because many words occur only once or twice in a training corpus it is impossible to make reliable statistical statements about them. This problem increases if we consider longer sequences of words as individual events in a statistical model.

Sub-word methods approach both these problems by splitting the words in a text into smaller character groups. The most common approach, the byte-pair encoding (BPE) algorithm [Sennrich & Haddow$^{+}$ 16c], splits all words in a text corpus down to sequences of characters. Then the most frequent character pair across all words is merged into a new token that replaces all occurrences of the underlying character pair in the text corpus. This process is repeated for a fixed number of merging operations, commonly around 5k-70k times. At the end of the process, all words are either merged completely or can be expressed as a sequence of relatively frequent sub-word units. The sequence of merge operations is stored as a BPE model, which can be applied to any text to obtain a consistent sub-word segmentation. Words that are unknown during training time can be expressed via these sub-words and, in the extreme case, spelled as sequences of individual characters. This means that sub-word segmentation approaches are a fluid middle ground between word-level and character-level methods, where an infinite number of merge operations yields a word-segmented corpus and zero merge operations achieve a character-level representation of the input text. Hence, byte-pair encoding acts as a bridge between an open vocabulary that can represent all possible words from a given alphabet and a system with a vocabulary that is technically closed.

A lower number of merge operations yields high sequence lengths of the text with low data sparsity and a small vocabulary. In contrast, a high number of merge operations returns a byte-pair encoding with shorter sequences from a large vocabulary that is exposed to higher data sparsity.

Several variations of the BPE algorithm are proposed, e.g. by incorporating morphological information [Huck & Riess$^{+}$ 17] or non-deterministic word segmentation [Kudo 18].

## 3.4 Evaluation Measures

Evaluating the quality of a translation hypothesis is an important step in building reliable, well-performing machine translation systems. While human evaluation is the gold standard of metrics, it is too costly and time-consuming to perform in practice during system development. Many automatic metrics have been developed, aiming to score translations comparably to human annotators. The shared task on metrics[1] of the WMT 2021 listed 15 metrics ranging from count-based to systems relying on neural network and word embeddings, working on a granularity ranging from word $n$-grams to characters [Freitag & Rei$^+$ 21].

Despite human judgment being the official metric of the WMT shared task on news translation, automatic evaluation measures have always played a crucial role. In this work, we rely on two long-time standard evaluation measures that allow comparison with both previous publications and state-of-the-art systems. However, starting in 2022 the WMT has been using Comet [Rei & Stewart$^+$ 20] as primary and ChrF [Popović 15] as secondary automatic metric [Kocmi & Bawden$^+$ 22], indicating a shift towards a new standard in automatic evaluation measures for the scientific community.

### 3.4.1 Bilingual Evaluation Understudy (BLEU)

The Bilingual Evaluation Understudy (Bleu) measures how many $n$-grams of a translation hypothesis $\hat{e}_1^{\hat{I}}$ can be explained through the reference [Papineni & Roukos$^+$ 02]. For many years it has been the de-facto gold standard for the evaluation of machine translation models, and despite much criticism [Callison-Burch & Osborne$^+$ 06, Mathur & Baldwin$^+$ 20, Freitag & Grangier$^+$ 20], the metric is used in almost every work on the matter. The core component of the Bleu metric is the *modified n-gram precision*

$$\mathrm{Prec}_n\left(e_1^I, \hat{e}_1^{\hat{I}}\right) := \frac{\sum_{w_1^n} \min\left\{\mathrm{count}(w_1^n\ ;\ e_1^I), \mathrm{count}(w_1^n\ ;\ \hat{e}_1^{\hat{I}})\right\}}{\sum_{w_1^n} \mathrm{count}(w_1^n\ ;\ \hat{e}_1^{\hat{I}})}$$

where the sum over $w_1^n$ considers each $n$-gram of the hypothesis $\hat{e}_1^{\hat{I}}$ once and $\mathrm{count}(w_1^n\ ;\ e_1^I)$ is defined as the number of times $w_1^n$ appears in $e_1^I$. Note that the denominator can be simplified to

$$\sum_{w_1^n} \mathrm{count}(w_1^n\ ;\ \hat{e}_1^{\hat{I}}) = \hat{I} - n + 1$$

i.e. the count of all $n$-grams in the hypothesis. Since precision-based metrics are prone to overvaluing short hypotheses, Bleu uses an explicit brevity penalty

$$\mathrm{BP}(I, \hat{I}) := \begin{cases} 1 & I \leq \hat{I} \\ e^{1-\frac{I}{\hat{I}}} & \text{else} \end{cases}$$

that reduces the score of short translations.

Sentence-level Bleu (Bleu$_s$) is computed as a geometric mean of four modified $n$-gram precisions multiplied with the brevity penalty

$$\mathrm{Bleu_s}\left(e_1^I, \hat{e}_1^{\hat{I}}\right) := \mathrm{BP}(I, \hat{I}) \cdot \sqrt[4]{\prod_{n=1}^{4} \frac{1}{4} \mathrm{Prec}_n\left(e_1^I, \hat{e}_1^{\hat{I}}\right)}.$$

Instead of Bleu$_s$ on the sentence level, the Bleu score is computed at the document level, that is, both the numerator and the denominator of the n-gram precision are accumulated over all sequences of the test set. Furthermore, it is common practice to report Bleu$^{[\%]}$ scores ranging from 0 to 100 by multiplying the ratio by a factor of 100. Since Bleu$^{[\%]}$ is a precision-based metric, higher Bleu scores indicate better translations.

[1] `https://www.statmt.org/wmt21/metrics-task.html`

### 3.4.2 Translation Edit Rate (TER)

The Translation Edit Rate (Ter) [Snover & Dorr+ 06] is an error metric based on the Levenshtein distance between a hypothesis and a reference translation. In the first step, the minimum number of edits needed to transform the hypothesis $\hat{e}_1^{\hat{I}}$ into the reference $e_1^I$ is computed. This distance is then divided by the length of the reference to obtain the Ter metric at the sentence-level

$$\text{Ter}_{\text{s}}\left(e_1^I, \hat{e}_1^{\hat{I}}\right) := \frac{\text{min number of edits to convert } \hat{e}_1^{\hat{I}} \text{ to } e_1^I}{I}.$$

Ter allows substitution, deletion, and insertion of words as well as shifts of word sequences as possible edits to convert $\hat{e}_1^{\hat{I}}$, and all edits are assigned equal costs.

Similar to Bleu, we report document-level $\text{Ter}^{[\%]}$ by accumulating the number of edits for each sentence pair of the dataset, dividing it by the accumulated hypothesis length and report the result as a percentage rather than as a ratio. Note that $\text{Ter}^{[\%]}$ is an error measure, meaning lower values refer to a better hypothesis, and that $\text{Ter}^{[\%]}$ scores of more than 100% are possible.

### 3.4.3 Perplexity

Language models are evaluated using perplexity (Ppl). This measurement is used in information theory to assess how well a probability distribution describes a series of data points. Perplexity is a transformation of the cross-entropy (see Equation 3.8)

$$\begin{aligned}\text{Ppl} = \text{Ppl}(p_\theta, \mathcal{D}) &:= \exp(\mathcal{L}_{\text{CE}}) \\ &= \exp\left(-\frac{1}{|\mathcal{T}|}\sum_{e_1^I \in \mathcal{D}} \ln p(e_1^I|\theta)\right)\end{aligned}$$

where $\mathcal{D}$ is the dataset on which the perplexity is reported.

Note that the perplexity is influenced by the underlying vocabulary in general and by the granularity of the input in particular. This means that sub-word perplexities (like BPE) are not comparable to word-level measurements.

# 4. Monolingual Data in Neural Machine Translation

Statistical machine translation requires text data from the languages involved. In the most common approach, bilingual, sentence-aligned data is used to train a machine translating system. This requires good-quality corpora with several hundreds of thousands of parallel sentence pairs. Looking at different data collections, such as Europarl[1], OpenSubtitles[2] or the data selection for the shared task on news translation of the Conference on Machine Translation (WMT)[3], it is evident that the amount of available bilingual data depends heavily on the language pair. An overview of the bilingual and monolingual data used in this work is given in Table 4.1.

Monolingual text data is an additional resource that is easier to obtain in practice and could provide important information about the structure of the languages involved. For English in particular, the bottleneck for monolingual data is computational power and possibly model capacity, not the amount of data available.

In recent years many works have investigated the training of *unsupervised* machine translation systems that rely purely on monolingual training data [Artetxe & Labaka$^+$ 18, Lample & Conneau$^+$ 18, Artetxe & Labaka$^+$ 19, Kim & Graça$^+$ 20]. In this work, however, we investigate how strong machine translation systems, trained on bilingual data, can benefit from additional monolingual text. We consider three different approaches: (1) We introduce an explicit language model into the training and search of the machine translation system, (2) we propose a multi-task training objective that relies on additional monolingual data and (3) we verify the results of the state-of-the-art method of back-translation [Sennrich & Haddow$^+$ 16b].

## 4.1 Language Model Fusion

Language models are a straightforward way to use monolingual data, and proved to be helpful in other language processing areas such as automatic speech recognition and handwriting recognition. In all state-of-the-art architectures for machine translation the decoder acts as a target-side language model predicting the upcoming target word $e_i$ provided the target history $e_0^{i-1}$. A drawback of this *implicit* or *internal language model* is that it requires either bilingual training data or special handling of monolingual data (see Section 4.3).

In this section, we discuss several approaches to combine an explicit language model $p_{\text{LM}}(e_1^I)$ with a machine translation model $p_{\text{TM}}(e_1^I|f_1^J)$. Since both the language model and the translation model provide a distribution over the target vocabulary for every target position $i$, we can combine them in a log-linear model

$$q(e_i; e_0^{i-1}, f_1^J) = p_{\text{TM}}^{\alpha}(e_i|e_0^{i-1}, f_1^J) \cdot p_{\text{LM}}^{\beta}(e_i|e_0^{i-1})$$

[1]`https://www.statmt.org/europarl/`

[2]`https://opus.nlpl.eu/OpenSubtitles-v2018.php`

[3]e.g. for the year 2020 `https://www.statmt.org/wmt20/translation-task.html`

Table 4.1: Overview of the amount of bilingual and monolingual data for the shared tasks used in this work. Tasks are provided by the IWSLT and the WMT.

| task | # words | | | |
|---|---|---|---|---|
| | source | | target | |
| | biling | mono | biling | mono |
| IWSLT En→It | 4.7M | 2.5G | 4.4M | 1.2G |
| WMT Ro→En | 16.2M | 54.8M | 15.9M | 1.4G |
| WMT De→En | 111.1M | 2.4G | 117.7M | 1.0G |
| WMT Zh→En | 99.9M | 117.2M | 389.0M | 1.0G |

where $\alpha$ and $\beta$ are the weights of the translation and language model respectively. Note that the log-linear combination requires that both models operate on the same target vocabulary $V_e$, which is particularly important if subword-level systems are employed.

The log-linear model combination $q$ of the two normalized models $p_{\text{TM}}, p_{\text{LM}}$ is not generally normalized. In order to obtain a normalized *fusion model* from $q$ we introduce an explicit renormalization which can be done on either the symbol or sequence level. In the following, we discuss symbol-level and sequence-level normalization strategies both for training and search.

### 4.1.1 Symbol-level Normalization

*Symbol-level* or *local normalization* is obtained by enforcing normalization of the fusion model at each target position $i$ over the target vocabulary $V_e$

$$\begin{aligned} p(e_1^I|f_1^J) &= \prod_{i=1}^{I} \frac{q(e_i; e_0^{i-1}, f_1^J)}{\sum_{\tilde{e}\in V_e} q(\tilde{e}; e_0^{i-1}, f_1^J)} \\ &= \prod_{i=1}^{I} \frac{p_{\text{TM}}^{\alpha}(e_i|e_0^{i-1}, f_1^J) p_{\text{LM}}^{\beta}(e_i|e_0^{i-1})}{\sum_{\tilde{e}\in V_e} p_{\text{TM}}^{\alpha}(\tilde{e}|e_0^{i-1}, f_1^J) p_{\text{LM}}^{\beta}(\tilde{e}|e_0^{i-1})}. \end{aligned}$$

This requires a sum over the target vocabulary $V_e$ for each target position $i$, resulting in $\mathcal{O}(I \cdot |V_e|)$ probabilities that need to be computed per sentence pair.

From Equation 3.6 we derive the training criterion

$$\begin{aligned} F_{\text{local}} &= \sum_{(f_1^J, e_1^I)\in\mathcal{T}} \log p(e_1^I|f_1^J) \\ &= \sum_{(f_1^J, e_1^I)\in\mathcal{T}} \sum_{i=1}^{I} \log \frac{p_{\text{TM}}^{\alpha}(e_i|e_0^{i-1}, f_1^J) p_{\text{t-LM}}^{\beta}(e_i|e_0^{i-1})}{\sum_{\tilde{e}\in V_e} p_{\text{TM}}^{\alpha}(\tilde{e}|e_0^{i-1}, f_1^J) p_{\text{t-LM}}^{\beta}(\tilde{e}|e_0^{i-1})} \end{aligned} \tag{4.1}$$

and from Equation 3.11a the decision rule with length normalization

$$\begin{aligned} \hat{e}_1^{\hat{I}} &= \hat{e}_1^{\hat{I}}(f_1^J) \\ &= \operatorname*{arg\,max}_{I, e_1^I} \left\{ \frac{1}{I} \sum_{i=1}^{I} \log \frac{p_{\text{TM}}^{\alpha}(e_i|e_0^{i-1}, f_1^J) p_{\text{s-LM}}^{\beta}(e_i|e_0^{i-1})}{\sum_{\tilde{e}\in V_e} p_{\text{TM}}^{\alpha}(\tilde{e}|e_0^{i-1}, f_1^J) p_{\text{s-LM}}^{\beta}(\tilde{e}|e_0^{i-1})} \right\} \end{aligned} \tag{4.2}$$

which is used to generate a hypothesis $\hat{e}_1^{\hat{I}}$. Note that different language models can be used during training and search, hence we distinguish between the language model $p_{\text{t-LM}}$ used in the training of the log-linear combination and $p_{\text{s-LM}}$, the language model applied during the search. As a special case, we consider the case where a language model is only introduced during the search, which is equivalent to choosing a training language model $p_{\text{t-LM}} = \frac{1}{|V_e|}$ with uniform distribution.

### 4.1.2 Sequence-level Normalization

*Sequence-level* or global normalization of a fusion model is achieved by normalizing over all possible target sequences

$$\begin{aligned}
p(e_1^I|f_1^J) &= \frac{q(e_1^I; f_1^J)}{\sum_{\tilde{I},\tilde{e}_1^{\tilde{I}}} q(\tilde{e}_1^{\tilde{I}}; f_1^J)} && (4.3a)\\
&= \frac{\prod_{i=1}^{I} p_{\text{TM}}^{\alpha}(e_i|e_0^{i-1}, f_1^J) p_{\text{LM}}^{\beta}(e_i|e_0^{i-1})}{\sum_{\tilde{I},\tilde{e}_0^{\tilde{I}}} \prod_{i=1}^{\tilde{I}} p_{\text{TM}}^{\alpha}(\tilde{e}_i|\tilde{e}_0^{i-1}, f_1^J) p_{\text{LM}}^{\beta}(\tilde{e}_i|\tilde{e}_0^{i-1})} && (4.3b)\\
&= \frac{1}{Z(f_1^J)} \prod_{i=1}^{I} p_{\text{TM}}^{\alpha}(e_i|e_0^{i-1}, f_1^J) p_{\text{LM}}^{\beta}(e_i|e_0^{i-1}). && (4.3c)
\end{aligned}$$

In the denominator, a sum over all target sequences $\tilde{e}_0^{\tilde{I}}$ is carried out, called the *normalization term* $Z(f_1^J)$. Restricting the target sequences to a maximum length of $I_{\max}$ yields a total of $\mathcal{O}(V_e^{I_{\max}})$ summands. This exponential number of computations prohibits sequence-level normalization in practice, where the target vocabulary $V_e$ easily reaches a size of 30k entries and sequences of length 25 or more are common, leading to $8 \times 10^{111}$ possible sequences. In the following, we discuss both the impact and possible solutions to this problem.

First, we consider the search problem for a sequence-level normalized model. Starting from the Bayes' decision rule (Equation 3.11a) we obtain

$$\begin{aligned}
\hat{e}_1^{\hat{I}} &= \underset{I,e_1^I}{\arg\max} \left\{ \sqrt[I]{p(e_1^I|f_1^J)} \right\}\\
&= \underset{I,e_1^I}{\arg\max} \left\{ \sqrt[I]{\frac{q(e_1^I; f_1^J)}{\sum_{\tilde{I},\tilde{e}_1^{\tilde{I}}} q(\tilde{e}_1^{\tilde{I}}; f_1^J)}} \right\}
\end{aligned}$$

where the denominator is a constant with respect to the maximization, hence we can omit it. Together with Equation 4.3 we get

$$\underset{I,e_1^I}{\arg\max} \left\{ \sqrt[I]{p(e_1^I|f_1^J)} \right\} = \underset{I,e_1^I}{\arg\max} \left\{ \frac{1}{I} \sum_{i=1}^{I} \log \left( p_{\text{TM}}^{\alpha}(e_i|e_0^{i-1}, f_1^J) p_{\text{s-LM}}^{\beta}(e_i|e_0^{i-1}) \right) \right\}$$

which does not rely on the computationally problematic sum over all target sentences. This makes the implementation of the search for sequence-level normalized models straightforward. Note that we can simplify the scaling factors in the exponent as well

$$\underset{I,e_1^I}{\arg\max} \left\{ \sqrt[I]{p(e_1^I|f_1^J)} \right\} = \underset{I,e_1^I}{\arg\max} \left\{ \frac{1}{I} \sum_{i=1}^{I} \log \left( p_{\text{TM}}(e_i|e_0^{i-1}, f_1^J) p_{\text{s-LM}}^{\frac{\beta}{\alpha}}(e_i|e_0^{i-1}) \right) \right\} \tag{4.4}$$

which can be controlled by a single parameter $\gamma := \frac{\beta}{\alpha}$.

Crucially, these simplifications cannot be applied during the training of the log-linear model.

The training criterion for sequence-level normalized models is

$$\begin{aligned}
F_{\text{global}} &:= \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log p(e_1^I | f_1^J) && (4.5) \\
&= \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log \frac{\prod_{i=1}^{I} p_{\text{TM}}(e_i | e_0^{i-1}, f_1^J)^{\alpha} p_{\text{t-LM}}(e_i | e_0^{i-1})^{\beta}}{\sum_{\tilde{I}, \tilde{e}_1^{\tilde{I}}} \prod_{i=1}^{\tilde{I}} p_{\text{TM}}(\tilde{e}_i | \tilde{e}_0^{i-1}, f_1^J)^{\alpha} p_{\text{t-LM}}(\tilde{e}_i | \tilde{e}_0^{i-1})^{\beta}} \\
&= \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \alpha \sum_{i=1}^{I} \log p_{\text{TM}}(e_i | e_0^{i-1}, f_1^J) \\
&\quad + \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \beta \sum_{i=1}^{I} \log p_{\text{t-LM}}(e_i | e_0^{i-1}) \\
&\quad - \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log \left( \sum_{\tilde{I}, \tilde{e}_1^{\tilde{I}}} \prod_{i=1}^{\tilde{I}} p_{\text{TM}}^{\alpha}(\tilde{e}_i | \tilde{e}_0^{i-1}, f_1^J) p_{\text{t-LM}}^{\beta}(\tilde{e}_i | \tilde{e}_0^{i-1}) \right).
\end{aligned}$$

However, the exact computation of the full denominator from Equation 4.3

$$Z(f_1^J) = \sum_{\tilde{I}, \tilde{e}_1^{\tilde{I}}} \prod_{i=1}^{\tilde{I}} p_{\text{TM}}^{\alpha}(\tilde{e}_i | \tilde{e}_0^{i-1}, f_1^J) p_{\text{t-LM}}^{\beta}(\tilde{e}_i | \tilde{e}_0^{i-1}) \tag{4.6}$$

is prohibitive in practice. Note that this is an infinite sum since $Z(f_1^J)$ sums over all possible length values $I$. In practice, we can ignore this and assume that for a given source sentence $f_1^J$ an upper target length $I_{\text{max}}$, beyond which there is no correct translation.

Even if we consider only a finite number of $\mathcal{O}(V_e^{I_{\text{max}}})$ target sequences, computing the normalization term $Z(f_1^J)$ is infeasible in practice. In the following, we discuss several possible approximations.

### $n$-best list approximation

Instead of computing the full sum over all possible sequences, we consider the subset of the most probable translations. We rely on an $n$-best list $\mathcal{B}_n := \mathcal{B}_n(p_{\text{TM}})$ generated from the not normalized log-linear model $q$ to approximate

$$\begin{aligned}
Z(f_1^J) &= \sum_{\tilde{I}, \tilde{e}_0^{\tilde{I}}} \prod_{i=1}^{\tilde{I}} p_{\text{TM}}^{\alpha}(\tilde{e}_i | \tilde{e}_0^{i-1}, f_1^J) p_{\text{LM}}^{\beta}(\tilde{e}_i | \tilde{e}_0^{i-1}) \\
&\approx \sum_{\tilde{e}_0^{\tilde{I}} \in \mathcal{B}_n} \prod_{i=1}^{\tilde{I}} p_{\text{TM}}^{\alpha}(\tilde{e}_i | \tilde{e}_0^{i-1}, f_1^J) p_{\text{LM}}^{\beta}(\tilde{e}_i | \tilde{e}_0^{i-1}).
\end{aligned}$$

This process is depicted in Figure 4.1. In practice, an $n$-best list $\mathcal{B}_n$, usually containing around 10-1000 sentences, is tiny compared to the full possible space of target sentences.

### Limited context assumption and trellis computation

Computing a sum over the scores of all possible sentences is a well-known problem in machine translation, going back to the IBM models [Brown & Pietra+ 93]. In the case of the IBM models, the full sum can be computed efficiently due to the very limited target context used by the

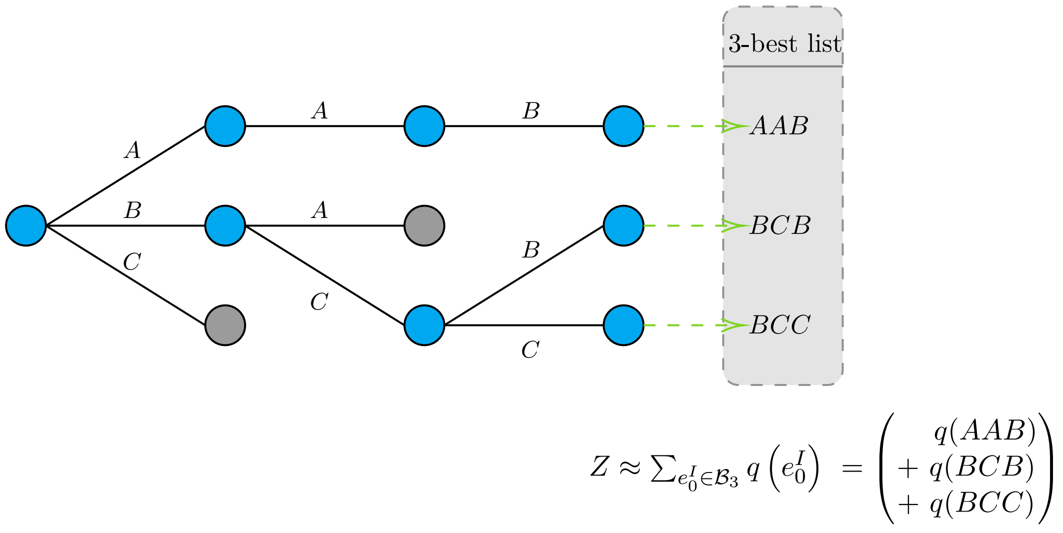

3-best list
A
B
C
A
A
C
B
B
C
AAB
BCB
BCC
$Z \approx \sum_{e_0^I \in \mathcal{B}_3} q\left(e_0^I\right) = \begin{pmatrix} q(AAB) \\ +\, q(BCB) \\ +\, q(BCC) \end{pmatrix}$

Figure 4.1: Example of a 3-best list generated from a model $q$ by a beam search with beam size 3. Nodes are colored gray to indicate that they are pruned from the search graph since none of their descendants are kept in the beam. To approximate the denominator the sum over the resulting 3-best list $\mathcal{B}_3$ is computed.

models. This stands in strong contrast to neural machine translation and language models, which in principle allow unlimited context size.

To get a better approximation of $Z(f_1^J)$

$$Z(f_1^J) = \sum_{\tilde{I}, \tilde{e}_0^{\tilde{I}}} \prod_{i=1}^{\tilde{I}} \underbrace{p_{\text{TM}}^{\alpha}(\tilde{e}_i|\tilde{e}_0^{i-1}, f_1^J) p_{\text{t-LM}}^{\beta}(\tilde{e}_i|\tilde{e}_0^{i-1})}_{=:q(\tilde{e}_i;\ \tilde{e}_0^{i-1}, f_1^J)}$$

for modern, neural-network-based models, we assume that only the last $k$ words of the translation are relevant to predict the next word

$$p_{\text{TM}}(e_i|e_0^{i-1}, f_1^J) \approx p_{\text{TM}}(e|e_{i-k+1}^{i-1}, \tilde{e}_0^{i-k}, f_1^J) \qquad \forall \tilde{e}_0^{i-k}.$$

This means we assume that the probability distribution for the current word $e_i$ depends only on the last $k$ target words and that any target context beyond this threshold does not affect the current probability distribution and is thus interchangeable. Since all relevant language and translation models have unlimited history, this generally does not apply; however, we can do an approximation.

$$p_{\text{TM}}(e_i|e_{i-k+1}^{i-1}, e_0^{i-k}, f_1^J) \approx p_{\text{TM}}(e_i|e_{i-k+1}^{i-1}, \hat{e}_0^{i-k}, f_1^J)$$

with

$$\hat{e}_0^{i-k} = \operatorname*{arg\,max}_{e_0^{i-k}} \left\{ p_{\text{TM}}(e_i|e_{i-k+1}^{i-1}, e_0^{i-k}, f_1^J) \right\}.$$

We apply the same approximation to the language model $p_{\text{LM}}$ and, for simplicity of notation, we drop the dependence on the source side on $q(e_i; e_0^{i-1}) = q(e_i; e_0^{i-1}, f_1^J)$.

Formally, we assume a model with a fixed context of size $k$ on the target sequence

$$q(e_i; e_0^{i-1}) = q(e_i; e_{i-k}^{i-1})$$

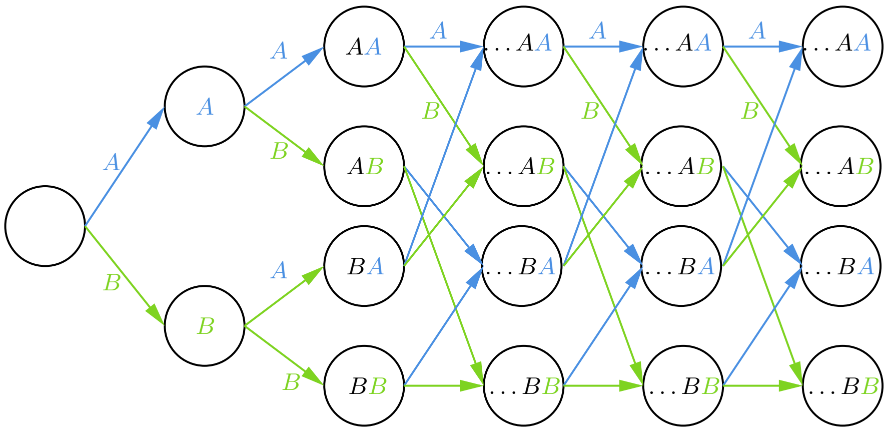


Figure 4.2: Example of a trellis with context $k = 2$ and vocabulary $V = \{A, B\}$.

We define the sum over the scores of all sequences of length $i$ that end with the subsequence $e_{i-k}^i$

$$Q(i, e_{i-k}^i) := \sum_{\tilde{e}_0^i : \tilde{e}_{i-k}^i = e_{i-k}^i} \prod_{i'=1}^{i} q(\tilde{e}_{i'}; e_{i'-k}^{i'-1}).$$

Since $q$ has a context of limited length $k$, we can compute the sum $Q(i, e_{i-k}^i)$ via dynamic programming. Starting from the definition of $Q$ we obtain

$$\begin{aligned} Q(i, e_{i-k}^i) &:= \sum_{\tilde{e}_0^i : \tilde{e}_{i-k}^i = e_{i-k}^i} \prod_{i'=1}^{i} q(\tilde{e}_{i'}; e_{i'-k}^{i'-1}) \\ &= q(e_i; e_{i-k}^{i-1}) \sum_{\tilde{e}_0^i : \tilde{e}_{i-k}^i = e_{i-k}^i} \prod_{i'=1}^{i-1} q(\tilde{e}_{i'}; e_{i'-k}^{i'-1}). \end{aligned}$$

where the last token $e_i$ is identical for all sequences in the sum

$$Q(i, e_{i-k}^i) = q(e_i; e_{i-k}^{i-1}) |V_e| \sum_{\tilde{e}_0^{i-1} : \tilde{e}_{i-k}^{i-1} = e_{i-k}^{i-1}} \prod_{i'=1}^{i-1} q(\tilde{e}_{i'}; e_{i'-k}^{i'-1}).$$

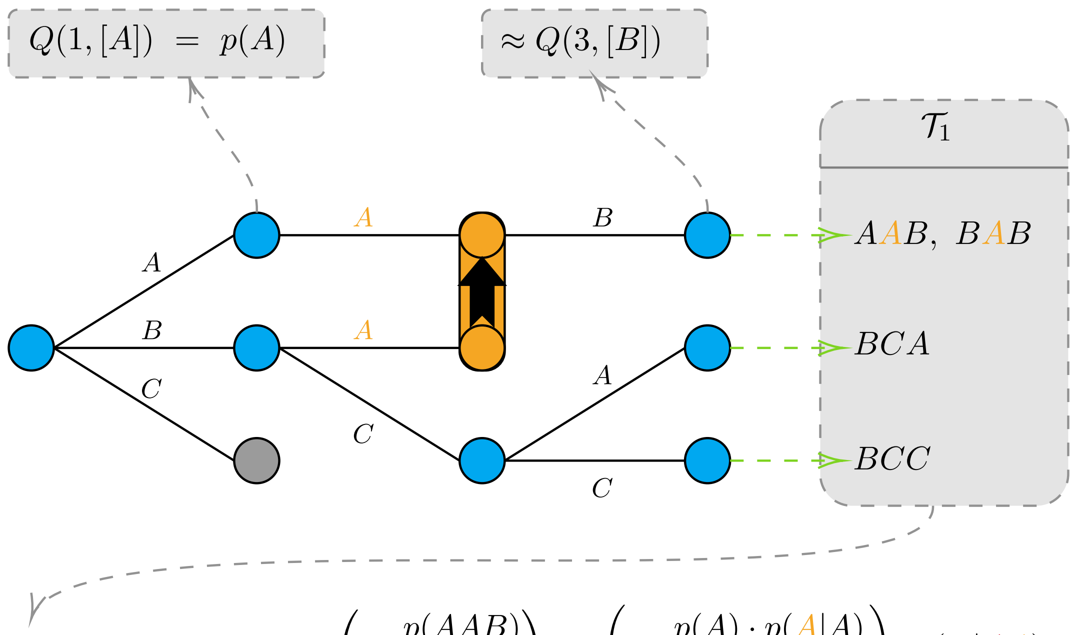
This can be formulated as a sum over all tokens at the position $i - k - 1$

$$\begin{aligned} Q(i, e_{i-k}^i) &= q(e_i; e_{i-k}^{i-1}) \sum_{e_{i-k-1} \in V_e} \left( \sum_{\tilde{e}_0^{i-1} : \tilde{e}_{i-k-1}^{i-1} = e_{i-k-1}^{i-1}} \prod_{i'=1}^{i-1} q(\tilde{e}_{i'}; e_{i'-k}^{i'-1}) \right) \\ &= q(e_i; e_{i-k}^{i-1}) \sum_{e_{i-k-1} \in V_e} Q(i-1, e_{i-k-1}^{i-1}). \end{aligned}$$

To compute $Q(i, e_{i-k}^i)$ a trellis can be constructed where each node of depth $i$ corresponds to a sequence $e_{i-k}^i$ and holds a $Q(i, e_{i-k}^i)$. An example of a trellis with $k = 2$ is depicted in Figure Figure 4.2, where each node represents a sequence ending with $e_{i-2}^i$.

Note that in practice, even with limited context, the exact computation of the sum over all sequences

$$Z_k = \sum_{I, e_{I-k}^I} Q(I, e_{I-k}^I).$$



$$Z \approx \sum_{e_0^I \in \mathcal{T}_1} p\left(e_0^I\right) = \begin{pmatrix} p(AAB) \\ +\, p(BAB) \\ +\, p(BCA) \\ +\, p(BCC) \end{pmatrix} \approx \begin{array}{l} \begin{pmatrix} p(A) \cdot p(A|A) \\ +\, p(B) \cdot p(A|B) \end{pmatrix} \cdot p(B|AA) \\ +\, p(B) \cdot p(C|B) \cdot p(A|BC) \\ +\, p(B) \cdot p(C|B) \cdot p(C|BC) \end{array}$$

Figure 4.3: Example of a trellis for a model with context $k = 1$ generated by a beam search with beam size 3. If two nodes of the same depth share a history of $k = 1$ tokens (highlighted in orange), the nodes are merged. Nodes are colored in gray to indicate that they are pruned from the search graph since none of their descendants are kept in the beam. To approximate the denominator $Z$ the sum over the resulting trellis $\mathcal{T}_1$ is computed. Approximations in the calculation are marked in red.

is still too costly, since at each time step $i$ it requires the storing and computation of each possible sequence ending $e_{i-k}^i$. Even for small vocabularies of 5,000 entries and a history size of 2, this is infeasible in practice.

Because creating the full trellis is impossible, we use a modified beam search procedure to create an approximation of the trellis by focusing on the most relevant target sentences. This *beam search with recombination* requires an additional parameter $k$ controlling the recombination history limit. The algorithm step closely resembles a traditional beam search where a list of $n$ partial hypotheses is iteratively expanded before being pruned according to the beam size $n$. In beam search with recombination, candidates for recombination are identified before the hypotheses are expanded. Candidates are selected for recombination if their last $k$ tokens are identical. If this is the case, these hypotheses are merged into a single beam entry by adding their scores and selecting the sequence with the highest score as the partial hypothesis. A visualization for history length $k = 1$ can be found in Figure 4.3.

In total, the trellis-based global normalization method relies on two approximations:

1. Instead of all possible sentences, the trellis is obtained from a beam search which prunes low-probability sequences.

2. Instead of using a 'true' limited context model, a full-context neural network model is applied, which relies on the most probable sentence that ends in the relevant context sequence.

### 4.1.3 Related Work

For a long time, the difference in availability between monolingual and bilingual data has made language model integration appealing. Historically, language models are included in the search procedure for machine translation systems, by applying the Bayes' theorem to the posterior probabilities such that the direction of the translation model is inverted [Brown & Cocke$^+$ 90]. Traditional phrase-based translation systems rely on language models as a crucial component to increase system performance [Brants & Popat$^+$ 07, Bojar & Chatterjee$^+$ 16]. In contrast to these, neural machine translation systems perform extremely well without an external language model [Bahdanau & Cho$^+$ 15, Vaswani & Shazeer$^+$ 17].

In neural machine translation systems, language model integration showed improvements on low-resource tasks by combining either the scores or the hidden states of the neural translation model and the language model [Gülçehre & Firat$^+$ 15, Gülçehre & Firat$^+$ 17]. These two approaches are often referred to as *shallow fusion* and *deep fusion* respectively. In contrast to the work described in this thesis, the training of the machine translation system and the language model are independent in both approaches and both models are only fused during the search process. Different score combinations with local re-normalization and joint training provided small improvements across several low-resource tasks [Stahlberg & Cross$^+$ 18]. While these methods showed improvements on low-resource and narrow domain datasets, they were not used in competitive neural machine translation systems [Bojar & Federmann$^+$ 18, Barrault & Bojar$^+$ 19].

To the best of our knowledge, no research has been published on globally normalized neural machine translation systems. The scarce existing work relies on approximations via 1-best lists [Murray & Chiang 18]. However, the underlying problem of summing over all possible target sequences arises similarly for the concept of minimum risk training. Recent work also relies on $n$-best lists to approximate the space of all sequences [Shen & Cheng$^+$ 16, Wu & Schuster$^+$ 16].

Globally normalized models are employed in other natural language processing tasks [Collobert & Weston$^+$ 11, Huang & Xu$^+$ 15, Andor & Alberti$^+$ 16] and automatic speech recognition [Michel & Schlüter$^+$ 20].

During the end-phase of this dissertation large language models [Brown & Mann$^+$ 20] gained a lot of attention inside and outside the scientific community. The training data for these models is typically several orders of magnitude bigger, compared to established language models. Early results show that these models are capable of translation [Brown & Mann$^+$ 20, Vilar & Freitag$^+$ 23] opening new ways to use language models and monolingual data for the task of machine translation.

## 4.2 Pre-Training and Multi-Task Learning

In this section, we consider the use of monolingual data to provide an auxiliary loss to the task of machine translation. Since state-of-the-art language and translation models are quite similar with respect to their architecture [Dai & Yang$^+$ 19, Irie & Zeyer$^+$ 19], it is possible to carry over the trained parameters of a language model to a translation model. The Bidirectional Encoder Representations from Transformers for Language Understanding (BERT) introduces another approach to train transformer-like architectures using only monolingual data by predicting missing words in a corrupted input sequence [Devlin & Chang$^+$ 19]. In this section, we propose and investigate different approaches using these monolingual models to improve a translation model. Instead of maintaining two separate models that are combined on the level of a probability distribution, we use the monolingual model to initialize certain parameters of the translation model. This *pre-training* approach can be extended to a *multi-task* model that is trained simultaneously on both translation and language modeling.

Formally, we consider a situation with two systems: a monolingual, and a translation model.

Each task requires the optimization of the corresponding training criterion with respect to a set of parameters on a set of training data. For the monolingual model $p_{\text{mono}}(e_1^I|\theta_{\text{mono}})$ we consider the training criterion $F_{\text{mono}}(\theta_{\text{mono}})$ with parameters $\theta_{\text{mono}}$ and monolingual training data $\mathcal{T}_{\text{mono}}$. Similarly, for translation the training criterion

$$F_{\text{TM}}(\theta) = \sum_{e_1^I \in \mathcal{T}} \log p(e_1^I|f_1^J, \theta).$$

is optimized with respect to the model parameters $\theta$.

Fundamentally, the application of either pre-training or multi-task learning methods requires that the model parameters are not disjoint $\theta \cap \theta_{\text{mono}} \neq \emptyset$.

We distinguish pre-training and multi-task learning approaches. Pre-training is a multi-stage approach consisting of:

1. Train monolingual model(s) and obtain $\theta^*_{\text{mono}}$ by optimizing $F_{\text{mono}}(\theta_{\text{mono}})$.
2. Transfer the language model parameters $\theta^*_{\text{mono}}$ to the translation model initialization $\theta^0$, initialize all remaining parameters randomly.
3. Train a translation model starting from $\theta^0$.

This pre-training approach relies on the fact that machine translation models are trained using an optimizer whose performance is based on the starting point $\theta^0$. Hence, parameter transfer from the monolingual model to the translation model is possible at initialization.

Multi-task approaches require a single training with a joint training criterion

$$F_{\text{multi}}(\theta_{\text{mono}}, \theta) = \sum_{e_1^I \in \mathcal{T}_{\text{mono}}} F_{\text{mono}}(\theta_{\text{mono}}; e_1^I) + \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log p(e_1^I|f_1^J, \theta),$$

where $\theta_{\text{mono}} \cap \theta \neq \emptyset$. Since the parameters are optimized on the two datasets $\mathcal{T}_{\text{mono}}$ and $\mathcal{T}$, iterative training procedures like SGD typically alternate update steps between the individual losses.

It is important to point out that the auxiliary loss function $F_{\text{mono}}$ can be defined on monolingual training data $\mathcal{T}_{\text{mono}}$ from either the source or the target language. This is a crucial advantage since all machine translation approaches that utilize monolingual data rely on monolingual target data and there is no established method to benefit from monolingual source data.

### 4.2.1 Pre-Training

#### Language Model Pre-Training

Transformer translation models rely on an encoder-decoder architecture. The encoder consists of several stacked bidirectional self-attention and feed-forward layers which transform the input source sentence $f_1^J$ into a sequence of hidden states $h_1^J$. The decoder of a transformer translation model operates rather similarly on the target sentence $e_0^I$. However, there are two important differences: i) the decoder self-attention layer is unidirectional and ii) an encoder-decoder cross-attention layer is added that allows the decoder to access the encoder states $h_1^J$. For a more detailed description of the transformer architecture, we refer the reader to Section 3.2.3. In the following, we split a transformer network into an encoder with parameters $\theta_{\text{enc-LM}}$ and a decoder with parameters $\theta_{\text{dec}} = \theta_{\text{dec-LM}} \dot{\cup} \theta_{\text{att}}$, where we denote the parameters of the cross-attention $\theta_{\text{att}}$ separately.

The notable difference between a transformer decoder and a language model is the cross-attention layer. For the task of language modeling there is no source sentence information, hence

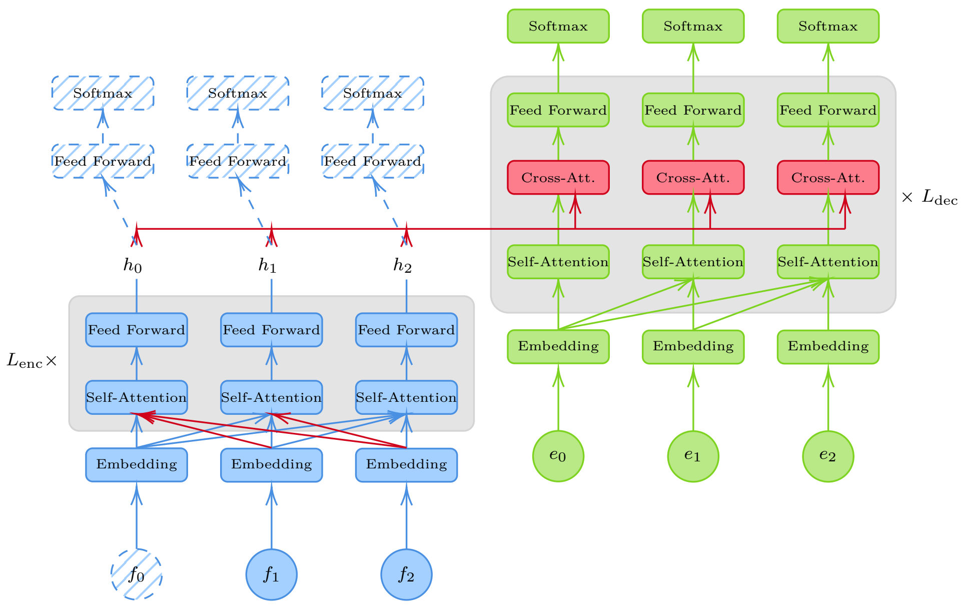


Figure 4.4: Schematic view of a transformer architecture. Highlighted layers compose a language model. Layers colored green denote a target-side language model in the decoder and blue components a source language model in the encoder, where dashed lines indicate components that are added to the encoder to create an encoder language model. Components highlighted in red are differences between the transformer and the corresponding language models.

both the encoder and the encoder-decoder cross-attention are omitted. However, this means that the vast majority of layers and parameters are arranged in the same way in translation decoders compared to language models. We denote this collection of these parameters $\theta_{\text{dec-LM}}$ to highlight that it only incorporates the language model part of the decoder and lacks the parameters of the cross-attention. The parameters of $\theta_{\text{dec-LM}}$ are marked in green in Figure 4.4 and are pre-trained via the training criterion

$$F_{\text{trg-LM}}(\theta_{\text{dec-LM}}) = \sum_{e_1^I \in \mathcal{T}_{\text{trg}}} \log p_{\text{trg}}(e_1^I | \theta_{\text{dec-LM}})$$

on the monolingual target data $\mathcal{T}_{\text{trg}}$.

The architectures of the encoder and the decoder of a transformer translation model are very similar. This means that after some small modifications the encoder can also be interpreted as a language model (see Figure 4.4). The main difference is the context available in the self-attention layer. In a translation model, the whole input sentence is available to the encoder, hence information from former and future input words is used at every source position $j$ in the bidirectional self-attention layer. In contrast to this, language models cannot access future words at positions $j' > j$, thus their architecture features unidirectional self-attention layers. Dropping all connections to future words allows for language model pre-training of the encoder. When employed in the full translation model, these connections are added again. Since the connections under consideration do not require additional parameters, this means that the full self-attention layer of the encoder is pre-trained. However, the parameters are used in a slightly different manner in the translation model. This mismatch will be discussed further later in this section.

In order to pre-train the encoder as a language model, we add a feed-forward projection layer

$W_{\text{proj}}$ and a softmax output layer on top of the last layer $L$

$$p_{\text{src}}(\bullet \,|f_1^{j-1}) = \text{softmax}(W_{\text{proj}} \cdot h_j^{(L)}) \in \mathbb{R}^{|V_f|}$$

and denote the additional parameters by $\theta_{\text{src-proj}}$ The resulting model $p_{\text{src}}$ is optimized according to the training criterion

$$F_{\text{src-LM}}(\theta_{\text{enc}}, \theta_{\text{src-proj}}) = \sum_{f_0^J \in \mathcal{T}_{\text{src}}} \log p_{\text{src}}(f_0^J|\theta_{\text{enc}}, \theta_{\text{src-proj}}). \tag{4.7}$$

To be consistent with language modeling approaches we also introduce a special token $f_0$ that marks the beginning of the source sentence. Note that the projection layer adds new parameters $\theta_{\text{src-proj}}$ to this source-side language model that are not part of the encoder.

Looking at the full parameters $\theta$ of the translation model, we observe that

$$\theta = \theta_{\text{dec-LM}} \,\dot{\cup}\, \theta_{\text{att}} \,\dot{\cup}\, \theta_{\text{enc}} \tag{4.8}$$

i.e. all parameters except the cross-attention are pre-trained. Next, we consider pre-training setups that allow training all parameters $\theta$ of the final translation model as well as reducing the architectural mismatch between training and pre-training.

While it is undeniably handy that language model pre-training can be applied equally to encoder and decoder, this generality comes with a drawback. By mapping both parts of the translation model to the same language model task, we reduce them to their lowest common denominator. For example, pre-training the cross-attention layers will not be possible with a generalized scheme, since these layers are not present in the encoder. This means that it is necessary to specialize the pre-training approaches to either the encoder or the decoder such that they better reflect the final architecture.

### Source Data Pre-Training for the Encoder Using BERT

The main difference between the architecture of an encoder and a language model is the direction of the self-attention (see Figure 4.4). An encoder can access the full input sentence using a bidirectional self-attention layer. However, adding information about future tokens in the input sequence makes the language modeling task trivial. In order to use monolingual data to train bidirectional architectures, the *masked language model task*, a variation of the language model task, and the Bidirectional Encoder Representations from Transformers for Language Understanding (BERT) was proposed [Devlin & Chang+ 19]. BERT is commonly used to train encoder-style architectures on a huge amount of monolingual data. The resulting model is either fine-tuned to a downstream task or used as a contextualized word embedding [Dou & Yu+ 19, Peters & Ruder+ 19, Zhang & Kishore+ 20]. Since machine translation models already produce strong word embeddings [Hill & Cho+ 15] that do not benefit from external training for realistic amounts of training data [Qi & Sachan+ 18], we fine-tune the BERT model for the downstream task of machine translation encoding rather than using BERT as a fixed, standalone word embedding. The BERT architecture is a very close adaptation of a transformer encoder, hence, all parameters of the encoder $\theta_{\text{enc}}$ are pre-trained.

In the following, we briefly describe the BERT architecture and training strategy as introduced by Devlin et al. [Devlin & Chang+ 19] and propose how to employ it in the training of translation models. So far we discussed encoder pre-training by using a source-side language model $p(f_j|f_0^{j-1})$. Instead of generating a sequence left-to-right, the BERT architecture aims to recover an input sequence $f_1^J$ from a distorted input $\tau(f_1^J)$ via the probability distribution

$$p_{\text{bert}}\left(f_1^J|\tau(f_1^J)\right) = \prod_{j=1}^{J} p_{\text{bert}}\left(f_j \mid j, \tau(f_1^J)\right).$$

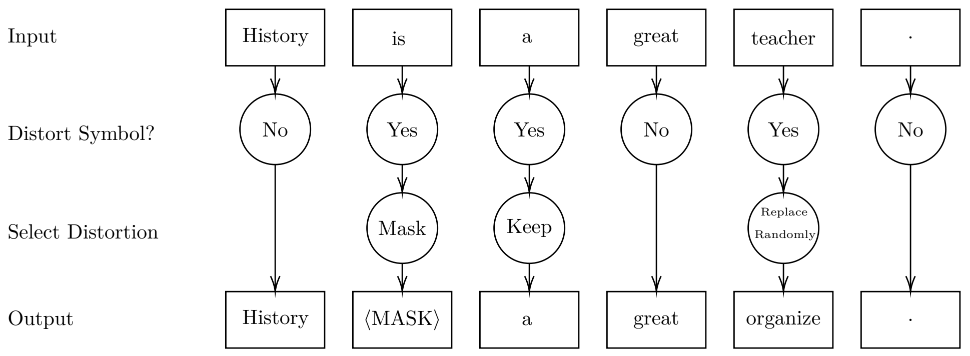


Figure 4.5: Example of the two-stage sequence distortion applied in BERT training. Example text is taken from newstest2014 of the WMT English→German task.

To generate a consistent sequence all words $f_1^J$ are generated from a series of shared hidden states. However, note that all $J$ positions are predicted simultaneously and that at position $j$ there is no dependence on previous words $f_{j'}$ with $j' < j$. This means that $p_{\text{bert}}$ is a non-autoregressive model, which limits its expressive power but allows for time-parallel training of the architecture. For transformer-style architectures this is very significant for the training speed.

BERT uses a two-stage distortion scheme $\tau$ in which each input token $f_j$ is considered independently. First, each input $f_j$ is randomly selected for distortion or left untouched. Formally, we sample from a Bernoulli distribution with parameter $\rho_{\text{distort}}$ for the events ‘DISTORT’ and ‘UNTOUCHED’ with

$$p_{\text{dist}}(X = \text{DISTORT}) = \rho_{\text{distort}}.$$

For each input $f_j$ we sample independently from this distribution and place it accordingly either in the set of untouched inputs $\mathbb{U}$ or the set of distorted inputs $\mathbb{D}$. This yields a partition of the input sequence $f_1^J$

$$\{1, \dots, J\} = \mathbb{U} \;\dot{\cup}\; \mathbb{D}.$$

Untouched inputs will not be changed by the distortion process i.e.

$$\tau(f_j) := f_j \qquad \forall j \in \mathbb{U}$$

and typically provide the majority of the inputs (the original author suggests an expected value of 85% untouched inputs [Devlin & Chang+ 19]).

Inputs $f_j$ selected in $\mathbb{D}$ are modified according a probabilistic noise function $p_{\text{noise}}$ in three different ways:

(i) with probability $\rho_{\text{mask}}$ the input is replaced by the symbol $\tau(f_j) := \langle\text{MASK}\rangle$

(ii) with probability $\rho_{\text{keep}}$ the input is unchanged $\tau(f_j) := f_j$

(iii) with probability $\rho_{\text{random}}$ the input is replaced with a (uniform) randomly selected word from the corresponding vocabulary $\tau(f_j) := v \sim V$

For an example of the distortion process, see Figure 4.5.

In the training criterion only positions selected for distortion $j \in \mathbb{D}$ are considered

$$F_{\text{bert}}(\theta_{\text{enc}}, \theta_{\text{src-proj}}) = \sum_{f_1^J \in \mathcal{T}_{\text{src}}} \sum_{j \in \mathbb{D}} \log p_{\text{bert}}(f_j \mid j, \tau(f_1^J); \theta_{\text{enc}}, \theta_{\text{src-proj}}), \tag{4.9}$$

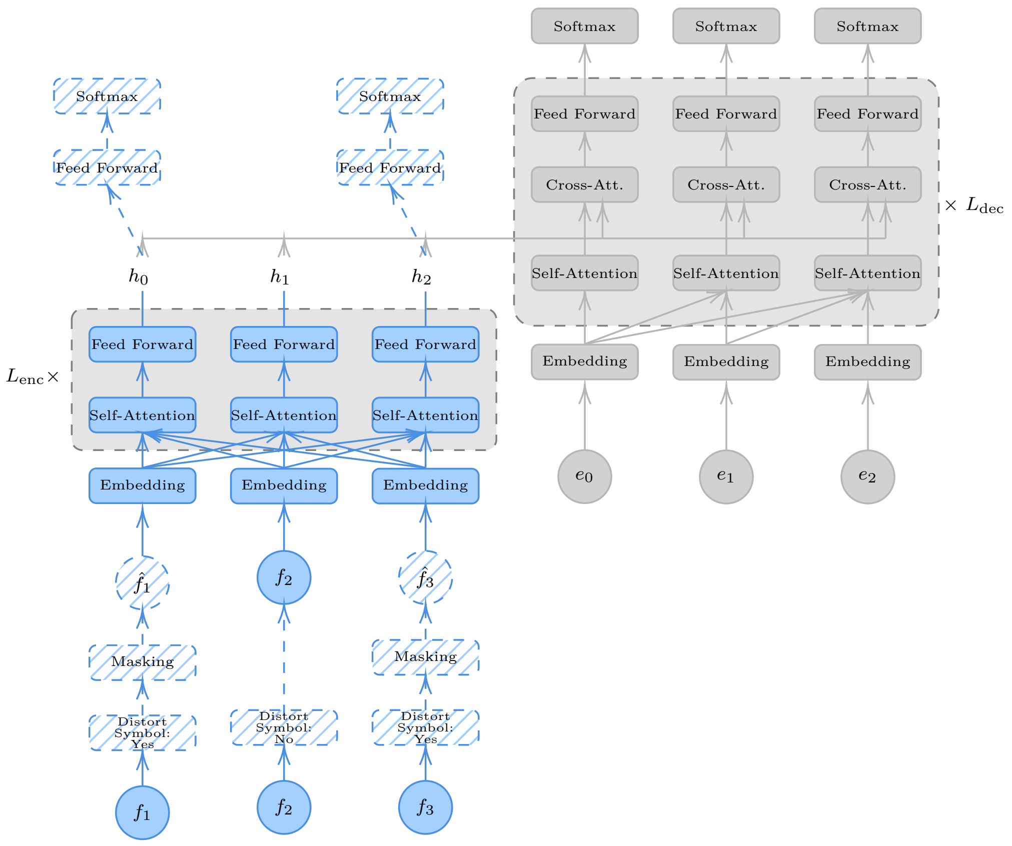


Figure 4.6: Encoder as BERT.

yielding a very important distinction between 'UNTOUCHED'positions $j \in \mathbb{U}$ and 'DISTORTED' positions $j \in \mathbb{D}$ with an unchanged input $f_j = \tau(f_j)$. The model is optimized to reconstruct the original token $f_j$ from the distorted input sequence $\tau(f_1^J)$. This training criterion $F_{\mathrm{bert}}$ can be interpreted as a form of *cloze loss*, named after a linguistic language completion test in which participants are asked to fill in the blanks created in a text by word deletion [Taylor 53].

Since untouched inputs are usually a significant majority $|\mathbb{U}| >> |\mathbb{D}|$, this means that

- the majority of the input sequence is correct and provides a meaningful context
- the model only gets feedback on a minority of its outputs, hence training is expected to be less efficient
- most positions considered in the loss do require a non-trivial correction of the input, which needs to rely on the context of the current word, but hardly the current word itself.

The final architecture as well as its place in a full transformer architecture are depicted in Figure 4.6. In particular, this approach trains all parameters of the encoder and mimics its bidirectional context.

Note that the distortion process is probabilistic, so if the same input $f_1^J$ is presented multiple times to the system (e.g. during different epochs in training or due to duplicates in the training data) the sequences $\tau(f_1^J)$ most likely differ over time. This means that, unlike language modeling, there is no probability for a certain sequence $f_1^J$. Probabilities are only estimated for an input sequence together with a certain distortion, and depending on whether important words are masked or kept in the input $f_1^J$, the reconstruction probability $p_{\mathrm{bert}}(f_1^J|\tau(f_1^J))$ may vary drastically.

### Target Data Pre-Training Using a Language Model Loss with Pseudo-Encoder

The biggest difference between a transformer language model and a translation model decoder is the existence of an encoder together with the cross-attention layers. When pre-training the decoder with monolingual target data, no source-side information is present, hence no encoder states $h_1^J$ are available. So far in decoder pre-training, we ignored the cross-attention layer and with it the encoder. This could be problematic as the cross-attention is the only component that provides source context, which is essential for any translation system. Hence, pre-training the decoder ignorant of the encoder might lead to a local optimum in which language model information is dominant and translation information is ignored. Furthermore, the lack of an attention mechanism creates a mismatch between the pre-training and the main training. During the main training most parameters of the decoder can be loaded from the pre-trained language model and the outputs of these layers can be considered reasonably expressive. However, the cross-attention parameters are initialized randomly, so at the beginning of the main training they purely add noise to the decoder, which raises the danger that the translation model learns simply to ignore the cross-attention output in its early epochs. In the past, different strategies considered several options to simulate an adequate encoder state [Sennrich & Haddow$^+$ 16b, He & Xia$^+$ 16].

We propose to use a pseudo-encoder with BERT-style masking to generate a series of hidden states which are used in the cross-attention layers. The pseudo-encoder operates on the target sentence, thus it provides information that is relevant to the decoder. However, if the full target sequence is handed to the pseudo-encoder, the decoder task of language modeling will become trivial. To avoid this, we use a BERT-style masking on the pseudo-encoder input together with a selective loss on the target positions. The pseudo-encoder operates on a distorted input sequence $\tau(e_1^I)$ which replaces the source sentence. Each input token is either placed in the set of distorted positions $\mathbb{D}$ or untouched positions $\mathbb{U}$, as described above. We obtain the pseudo-encoder output similar to Equation 3.1 as

$$\hat{h}_1^I := \textsc{Encoder}\left(\tau(e_1^I)\right).$$

All outputs $\hat{h}_1^I$ of the pseudo-encoder are accessible in the cross-attention. However, only the set of distorted positions $\mathbb{D}$ is considered in the cross-entropy loss to stop the model from overproducing copies of the input. This yields the training criterion for pre-training

$$F_{\text{pseudo-enc}}(\theta) = \sum_{e_1^I \in \mathcal{T}_{\text{trg}}} \sum_{i \in \mathbb{D}} \log p_{\text{bert}}(e_i \mid i, e_0^{i-1}, \tau(e_1^I); \theta).$$

In this pre-training scheme, the decoder accesses the full target sequence up to the position $i-1$ internally and obtains information about future words through the pseudo-encoder and the cross-attention layer.

The full pre-training architecture is depicted in Figure 4.7. Note that the entire transformer can be pre-trained this way and no additional parameters have to be added. Only the masking and the loss function differ from a standard transformer. Hence, after pre-training the transformer this way, it is possible to reuse all parameters directly in a translation model. However, if another pre-training is applied to explicitly train the encoder on source data, we disregard the parameters from the pseudo-encoder and instead initialize the translation model encoder with the specialized encoder.

## 4.2.2 Multi-Task Learning

All approaches discussed so far can be applied in pre-training as well as in multi-task learning. In pre-training approaches, the training for the supportive language modeling task is completed

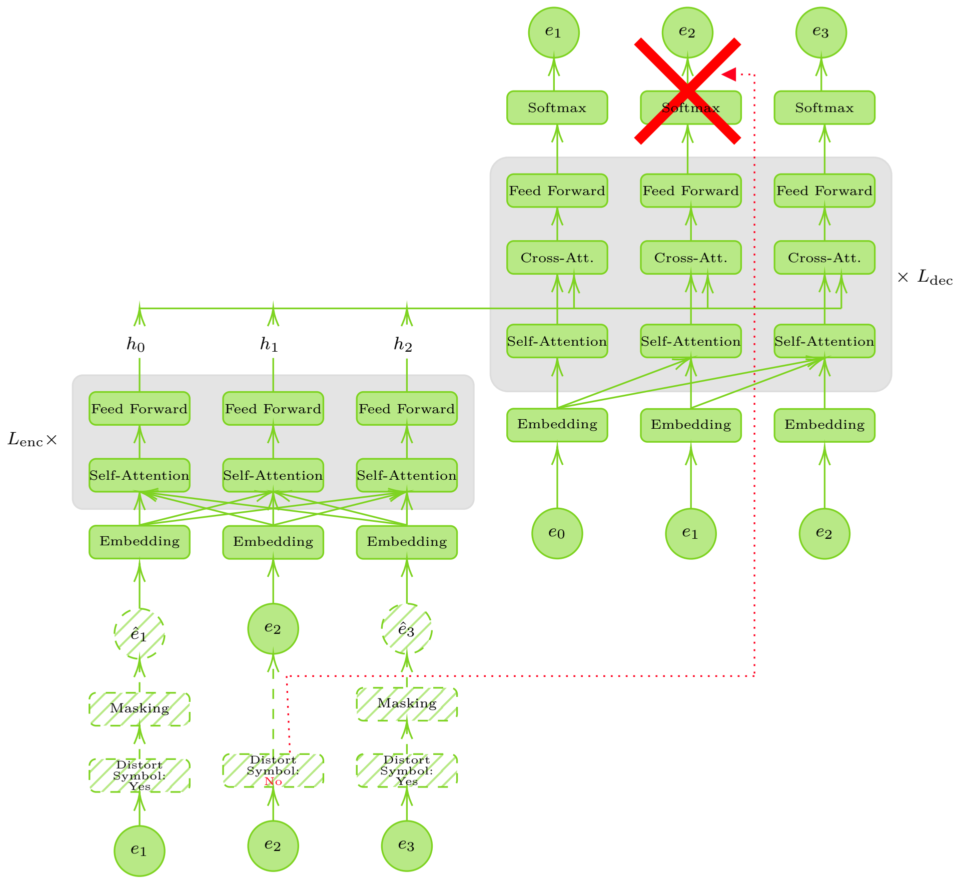


Figure 4.7: Schematic view of a transformer architecture interpreted as decoder language model with pseudo-encoder. Dashed lines indicate components that are not part of a transformer architecture. In this example $\mathbb{D} = \{1, 3\}$ and $\mathbb{U} = \{2\}$. Token $e_2$ is not predicted during training and will not be part of the loss as it is not distorted.

first and the resulting parameters are used as an initialization point for the main training of the translation model. If the optimization algorithm of the main training moves sufficiently far in the parameter space, the impact of the initialization point decreases. This means that the parameters of the final model might not be suitable for the pre-training task, a phenomenon typically called *catastrophic forgetting*. In multi-task translation, the language modeling (or BERT) objective is optimized in parallel with the translation model objective. Since each optimization step has to balance both objective functions, the multi-task trained model should not suffer from catastrophic forgetting.

All four pre-training approaches presented are suitable for multi-task training with straightforward multi-task training criteria

- Encoder language model:

$$F(\theta) = \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log p(e_1^I | f_1^J; \theta) + \sum_{f_1^J \in \mathcal{T}_{\text{src}}} \log p_{\text{src}}(f_1^J | \theta_{\text{enc}}, \theta_{\text{src-proj}}) \tag{4.10}$$

- Decoder language model:

$$F(\theta) = \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log p(e_1^I | f_1^J; \theta) + \sum_{e_1^I \in \mathcal{T}_{\text{trg}}} \log p_{\text{trg}}(e_1^I | \theta_{\text{dec-LM}}) \tag{4.11}$$

- Encoder BERT model:

$$F(\theta) = \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log p(e_1^I | f_1^J; \theta) + \sum_{f_1^J \in \mathcal{T}_{\text{src}}} \sum_{j \in \mathbb{D}} \log p_{\text{bert}}(f_j \mid j, \tau(f_1^J); \theta_{\text{enc}}, \theta_{\text{src-proj}}) \tag{4.12}$$

- Decoder language model with pseudo-encoder:

$$F(\theta) = \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log p(e_1^I | f_1^J; \theta) + \sum_{e_1^I \in \mathcal{T}_{\text{trg}}} \sum_{i \in \mathbb{D}} \log p_{\text{bert}}(e_i \mid i, e_0^{i-1}, \tau(e_1^I); \theta). \tag{4.13}$$

In all cases the decision rule of the translation model is unaffected.

### 4.2.3 Related Work

Pre-training and multi-task learning are well-established methods in the training of artificial neural networks [Pratt & Mostow+ 91, Caruana 93]. Even before neural networks achieved state-of-the-art performance in machine translation, there were investigations to introduce these methods to natural language tasks [Collobert & Weston 08, Collobert & Weston+ 11]. In particular, the rise of expressive word embeddings provided a powerful, general-purpose component to include across models for tasks with low training data [Mikolov & Chen+ 13, Pennington & Socher+ 14]. The introduction of contextualized word embeddings from language models [Peters & Neumann+ 18] led to a rise in investigations regarding the adaption of pre-training models to natural language processing tasks [Gardner & Grus+ 18, Qiu & Sun+ 20]. In 2019 the Bidirectional Encoder Representations from Transformers for Language Understanding (BERT) provided a new way to train contextualized word embeddings based on a transformer encoder [Devlin & Chang+ 19]. This sparked a wave of papers both adapting pre-training or transfer learning based on the BERT architecture [Dou & Yu+ 19, Peters & Ruder+ 19, Zhang & Kishore+ 20] as well as variations of the model [Zhang & Han+ 19, Yang & Dai+ 19, Clark & Luong+ 20, Lan & Chen+ 20, Liu & Gu+ 20].

However, BERT and other word embeddings are commonly applied to natural language understanding tasks where labeled data is scarce. This is oftentimes not the case for machine translation and raises the question of whether pre-training is a viable strategy in machine translation, and this is the focus of our work. Using context-free word embeddings does not improve machine translation models if reasonable amounts of bilingual training data are available [Qi & Sachan$^+$ 18], and the embedding layer of a neural machine translation system already produces strong and meaningful word embeddings [Hill & Cho$^+$ 15]. Hence, we focus on pre-training more complex and extended parts of the architecture. The idea of pre-training parts of the encoder and decoder as language models brought improvements over the use of back-translation for the LSTM-based architectures and laid the foundation for this work [Ramachandran & Liu$^+$ 17]. Parallel to our investigation, Lample et al. applied very similar strategies to pre-train a transformer translation model as language or BERT model [Conneau & Lample 19], and we discuss their results in Section 4.4. Several works investigate approaches to pre-train a transformer architecture with encoder and decoder for various downstream tasks [Lewis & Liu$^+$ 20, Raffel & Shazeer$^+$ 20, Liu & Gu$^+$ 20]. A very popular branch of multi-task learning in machine translation is the use of multilingual systems [Ha & Niehues$^+$ 16]. Multilingual machine translation systems aim to translate between several language pairs. Learning the translation for each language pair can be viewed as a sub-task of a multi-task learning framework. Since these tasks can all be solved with the same architecture and loss function, state-of-the-art approaches simply mix the training data for all language pairs and provide a special symbol that indicates the language direction [Johnson & Schuster$^+$ 17, Aharoni & Johnson$^+$ 19]. Multilingual systems train all languages in parallel; however, in the domain of low-resource translation, pre-training (often called transfer learning) is frequently applied to boost weak systems with parallel data from related language pairs [Zoph & Yuret$^+$ 16, Kocmi & Bojar 18, Kim & Gao$^+$ 19, Haddow & Bawden$^+$ 22].

## 4.3 Back-Translation

For the training of machine translation models, bilingual training data is naturally the most important resource. For many language pairs, monolingual target data $\mathcal{T}_{\text{trg}}$ is widely available and it is commonly used via *back-translation* [Sennrich & Haddow$^+$ 16b]. Back-translation is the well-established state-of-the-art using monolingual target data and is employed in countless submissions to shared translation tasks [Sennrich & Haddow$^+$ 16a, Schamper & Rosendahl$^+$ 18, Stahlberg & de Gispert$^+$ 18, Zhou & Zhou$^+$ 21, Jon & Popel$^+$ 22].

The idea of back-translation is to create a bilingual corpus from monolingual target data by adding a machine-generated or synthetic source sentence to each monolingual target sentence. In the first step, a *back-translation model* $p_{\text{trg}\to\text{src}}$ is trained using the same architecture and training data as the original translation model; however, the sides of the training corpus $\mathcal{T}$ are flipped

$$\mathcal{T}_{\text{bt-model}} := \left\{(e_1^I, f_1^I) \mid (f_1^I, e_1^I) \in \mathcal{T}\right\}.$$

After training the back-translation model $p_{\text{trg}\to\text{src}}$, the monolingual target data is translated

$$\hat{f}_1^{\hat{J}} := \arg\max_{J, f_1^J}\{p_{\text{trg}\to\text{src}}(f_1^J|e_1^I)\}$$

and combined into a synthetic training corpus

$$\mathcal{T}_{\text{bt}} := \left\{(\hat{f}_1^{\hat{J}}, e_1^I) \mid e_1^I \in \mathcal{T}_{\text{trg}}\right\}.$$

The training of the final (forward) translation model $p$ is then performed using the joint training data $\mathcal{T} \cup \mathcal{T}_{\text{bt}}$. Back-translation uses the fact that translation is a symmetric task, i.e. switching the input and the output of the task does not fundamentally change it. Because of this, the approach is translation-specific and cannot be applied as a general semi-supervised method in other tasks such as automatic speech recognition or text summarization.

Back-translation generates a corpus consisting of synthetic source sentences and human-generated target sentences. This approach is motivated by the intuition that the decoder of a translation model contains an implicit language model that benefits from a stronger training signal. Back-translation can be interpreted as a multi-task training approach. Multi-task training the decoder as a translation and language model allows the use of monolingual and bilingual training data in the optimization process of the translation model (see Section 4.2). A problem when feeding monolingual target data to the encoder is the lack of a source-side input. The lack of an encoder output provides a challenge from a technical point of view since the model is not defined if there is no encoder. In the methods presented in this work, we approach this problem by removing the cross-attention layer, essentially setting the encoder outputs to zero. This means that neither the encoder nor the cross-attention is pre-trained and the pre-training optimizes the decoder to only focus on target-side information. We extended this approach and added a pseudo-encoder which reads a distorted version of the source to simulate encoder outputs. In 2015 Sennrich et al. investigated different methods of simulating a meaningful encoder state for monolingual target data [Sennrich & Haddow⁺ 16b]. The best performance was achieved by back-translation, i.e. when the monolingual target sentence $e_1^I$ was provided with a synthetic source sentence $\hat{f}_1^J$. Overall, we can formulate back-translation as multi-task training where the encoder state (Equation 3.1) is modified

$$h_1^J := \textsc{Encoder}\left(\underset{J,f_1^J}{\arg\max}\left\{p_{\text{trg}\rightarrow\text{src}}(f_1^J|e_1^I)\right\}\right).$$

This multi-task interpretation can be made explicit by marking the synthetic and original data in training [Caswell & Chelba⁺ 19].

### 4.3.1 Related Work

Back-translation, i.e. the use of synthetic training data with the target sentences written by humans and the source sentences generated by a machine translation system, was introduced in 2008 [Schwenk 08] and later adapted to neural machine translation in 2015 [Sennrich & Haddow⁺ 16b]. It quickly became part of most state-of-the-art systems and remains the strongest and most commonly used way to incorporate monolingual data into machine translation systems [Barrault & Bojar⁺ 19, Barrault & Biesialska⁺ 20, Akhbardeh & Arkhangorodsky⁺ 21].

While the approach in principle is independent of the model architecture, back-translation is typically applied in neural machine translation systems. Phrase-based translation systems, on the other hand, frequently rely on language models to incorporate the monolingual training data [Koehn & Hoang⁺ 07, Wuebker & Huck⁺ 12, Bojar & Buck⁺ 14], with few works investigating the effect of synthetic sources as an alternative [Bojar & Tamchyna 11].

Conceptually, back-translation is a straightforward approach and many improvements have been suggested over the years. The *dual learning* approach relies on the symmetry of forward and backward translation, i.e. that the tasks of source→target and target→source translation only differ in the data used. In this approach, a forward and a backward translation system are trained on the respective datasets and a segment of back-translation data is used to update the forward translation system. This system in turn can generate *forward translation* sentence pairs by translating source sentences, which can be used as training data for the back-translation system. The training loss of these dual models is augmented by a source and target language model that reward well-formed sentences [He & Xia⁺ 16].

A simpler variation of back-translation is the use of copied data pairs. Instead of translating a target sentence into the source language to obtain a new training pair, the target sentence is simply copied. This creates a trivial target-target data pair that can be used to train the machine translation systems in a low-resource scenario [Currey & Miceli Barone$^+$ 17].

A common criticism of back-translation approaches is the lack of linguistic diversity in the resulting data. Machine translation systems tend to generate target sentences with less morphological variety compared to human translators. Different decoding approaches can lead to improved back-translation data with more diverse translations [Edunov & Ott$^+$ 18, Imamura & Fujita$^+$ 18, Graça & Kim$^+$ 19].

The difference in linguistic variety leads to an interpretation that considers the training on back-translated data as a separate task in a multi-task learning setup. Similar to a multilingual system, each sentence pair in training is marked by a special tagging symbol to distinguish original bilingual data from back-translated [Caswell & Chelba$^+$ 19].

## 4.4 Experimental Results

In the following section, we present the results of the experimental evaluation of the methods discussed here. We start by describing the general setup, including the data conditions and the hyperparameters for all experiments in this thesis.

### 4.4.1 Experimental Setup

We evaluate the presented methods on several openly available datasets of different languages and data sizes. The smallest training corpus is the English→Italian data of the shared task on multilingual machine translation[4] of the International Workshop on Spoken Language Translation (IWSLT) 2017 [Cettolo & Federico$^+$ 17]. The training corpus consists of English TED talks[5] together with their human translation to Italian [Cettolo & Girardi$^+$ 12]. This yields a corpus of 4.4M Italian words. Since the corpus originates from English text, there is no in-domain Italian target data. Hence, we use monolingual news crawl data for both English and Italian[6]. We report results on the development set `dev2010` and use `tst2010` and `tst2017` as test sets.

For Romanian→English we follow the low-resource shared task on news translation[7] of the Conference on Machine Translation (WMT) 2016 [Bojar & Chatterjee$^+$ 16]. The data is collected from a government-sponsored news website[8] and from speeches in the European parliament. As development set we use `updated_newsdev2016` and as test set `newstest20116`. Monolingual data is provided for all WMT shared tasks on news translation in the form of `News Crawl` corpora. These monolingual data collections originate from news articles in the respective languages and thus match the domain of the corresponding test sets.

Furthermore, we report results on two high-resource tasks, namely the German→English and Chinese→English shared translation tasks[9] of the WMT 2018.

The German→English task uses datasets collected from websites, parliamentary speeches and press releases as well as news comments. With 117M running target words, it is significantly bigger than the first two tasks. We use the monolingual training data obtained from the `News Crawl 2017-2018` corpora for both source and target. Selecting `newstest2015` as the development set, we report final results on the three test sets `newstest2014`, `newstest2017` and `newstest2018`.

Chinese→English is the biggest task we consider, with 389M target words. The majority of the training data stems from parliamentary documents of the United Nations with a small selection

[4]`https://workshop2017.iwslt.org/index.php`
[5]`https://www.ted.com/`
[6]`https://data.statmt.org/news-crawl/it/`
[7]`https://www.statmt.org/wmt16/translation-task.html`
[8]The ‘Southeast European Times’, which stopped publications in 2015
[9]`https://www.statmt.org/wmt18/translation-task.html`

of news and commentaries. Similar to the other WMT tasks, we use the `News Crawl` corpus as source for monolingual training data, selecting all available Chinese corpora on the source side and the English `News Crawl 2017-2018` datasets as monolingual target data. The bilingual data is filtered and 17M sentence pairs are extracted using uniblock [Gao & Wang$^+$ 19]. During training, we use `newsdev2017` as the development set and we report final results on `newstest2017` and `newstest2018`.

Table 4.2: Overview over the amount of bilingual and monolingual training data.

| task | data type | number of words | |
|---|---|---|---|
| | | source | target |
| En→It | bilingual | 70.7k | 103.6k |
| | monolingual | 2.5G | 1.2G |
| Ro→En | bilingual | 16.2M | 15.9M |
| | monolingual | 54.8M | 1.4G |
| De→En | bilingual | 111.1M | 117.7M |
| | monolingual | 2.4G | 1.0G |
| Zh→En | bilingual | 99.9M | 389.0M |
| | monolingual | 117.2M | 1.0G |

An overview over the amount of bilingual and monolingual training data for the four tasks is given in Table 4.2. For a detailed overview of the training, development and test data for the four tasks, we refer the reader to Appendix A.1.

All experiments are performed with a transformer machine translation model as described in Section 3.2.3. Our models are implemented in the RETURNN toolkit [Doetsch & Zeyer$^+$ 17], which is based on TensorFlow [Abadi & Agarwal$^+$ 15]. If not specified, the hyperparameters of the model are chosen according to the 'base' setup from the original publication [Vaswani & Shazeer$^+$ 17]. In particular, we use $L = 6$ layers in both the encoder and the decoder with $N = 8$ heads in each attention layer. Most layers use a hidden size of $d_{\text{model}} = 512$ except for the inner state of the feed-forward block with $d_{\text{ff}} = 2048$. We use the pre-norm approach of layer normalization as described in Section 3.2.3. The number of training epochs, dropout and the vocabulary size are chosen according to the task. The size of the vocabulary has a strong impact on the model size and consequently the amount of GPU memory used. Depending on the amount of memory remaining, we choose the biggest possible batch size for each task and employ gradient accumulation to stabilize the training. The task-specific values are listed in Table 4.3.

We run each experiment on a single GPU using the Adam optimizer [Kingma & Ba 15] with default parameters. Additionally, we scale the learning rate by a factor of 0.7 to 0.9 if the cross-entropy loss on the development set does not improve for several consecutive checkpoints. No warm-up training steps are used and we select the best model based on the development set Bleu score.

Language models are built and trained using best practices laid out by Irie and Zeyer [Irie & Zeyer$^+$ 19] with some small variations. Due to modeling and GPU memory constraints, all models use 6 layers. Furthermore, we use the vocabulary obtained from a translation model, which in some setups is shared across the source and target language. We consider two different language model setups: (a) as a standalone model and (b) as pre-training model for a machine translation system. Standalone language models follow Irie and Zeyer [Irie & Zeyer$^+$ 19] using a standard SGD algorithm with learning rate scheduling and a post-norm approach for layer normalization in the sub-layers. In this setup we use $d_{\text{model}} = 1024$ and $d_{\text{ff}} = 4096$. Language models that are part of a pre-training setup need to be compatible with the final translation model. Hence, we use $d_{\text{model}} = 512$ and $d_{\text{ff}} = 2048$, a post-norm approach for layer normalization and apply the

Table 4.3: Task-specific parameters of the transformer translation model.

| task | BPE | | effective batch | dropout |
|---|---|---|---|---|
| | #ops | joint? | size (tokens) | |
| En→It | 8k | no | 18k | 0.3 |
| Ro→En | 20k | yes | 14k | 0.2 |
| De→En | 50k | no | 28k | 0.1 |
| Zh→En | 50k | no | 18k | 0.1 |

Table 4.4: Perplexity of the target-side language models across all tasks. Language models are trained using either the target side of the monolingual data only or the combination of this with monolingual target data.

| language pair | lm trained on | Ppl (BPE) | |
|---|---|---|---|
| | | dev | test |
| En→It | biling-trg | 53.4 | 49.3 |
| | + mono | 34.4 | 33.9 |
| Ro→En | biling-trg | 115.9 | 113.6 |
| | + mono | 29.2 | 29.1 |
| De→En | biling-trg | 84.0 | 91.7 |
| | + mono | 39.7 | 51.6 |
| Zh→En | biling-trg | 81.3 | 88.1 |
| | + mono | 44.4 | 36.1 |

Adam optimizer. Note that both variations of language model have the same amount of layers and we observe very similar performance with respect to perplexity.

All text data is pre-processed using the tokenizer and punctuation normalizer from the Moses toolkit [Koehn & Hoang$^+$ 07]. The resulting text is split into sub-words via a byte-pair encoding [Sennrich & Haddow$^+$ 16c], and for the Romanian text we remove all diacritics.

All systems are evaluated using case-sensitive, tokenized, word-level BLEU scores[10] from the SacreBLEU toolkit [Post 18]. Furthermore, we report Ter scores for all experiments, calculated by the TERCom toolkit[11].

### 4.4.2 Language Model Fusion

In this section, we report our empirical findings on language model fusion. First, we investigate whether monolingual data in general and language models in particular are able to provide meaningful information to the translation process. Next, we study the impact of adding a language model and performing different re-normalization strategies during the search. In the final step, we consider ensembles of language and translation models that are trained jointly.

To analyze the impact of monolingual target data, we consider two types of language models. The first type is trained on all target data available, in particular a big corpus of monolingual data and the target side of the bilingual data. The second type emulates the internal language model of a translation model decoder and is trained only on the target side of the bilingual data. In Table 4.4 we report the perplexities for both language model types and observe that adding monolingual data to the language model training roughly halves the perplexity on the development set. Only for English→Italian do we we observe a smaller, yet still significant, reduction to 65%. This seems counterintuitive since English→Italian is the smallest of all four tasks, providing the

[10]SacreBLEU Signature: `BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.5.0`
[11]`http://www.cs.umd.edu/~snover/tercom/`

lowest amount of bilingual target data while maintaining a comparable amount of monolingual data for the language model. However, English→Italian is the only task where the domain of the monolingual data (news) does not match the domain of the development and test set. As WMT tasks, Romanian→English, German→English and Chinese→English provide a test set from the news domain, while the IWSLT task English→Italian is evaluated on transcribed TED talks. Overall, we observe a great improvement in language model performance when monolingual data is added. We conclude that monolingual data provides information on the target language which a neural network model cannot easily extract from the target data seen in training. Since a language model and a translation model decoder are almost identical in architecture, loss and training procedure, we conclude that monolingual data is a resource that can help to improve state-of-the-art translation models. In the following, we consider how this can be achieved by introducing an explicit language model via log-linear model combination.

Table 4.5: Impact of reducing the amount of decoder layers from 6 to 2 across all language pairs. All results are reported on the development set.

| language | #layers | | dev | |
|---|---|---|---|---|
| pair | enc | dec | Bleu [%] | Ter [%] |
| En→It | 6 | 6 | 28.1 | 54.0 |
| | | 2 | 26.9 | 55.0 |
| Ro→En | 6 | 6 | 35.7 | 45.3 |
| | | 2 | 33.0 | 48.1 |
| De→En | 6 | 6 | 32.7 | 48.6 |
| | | 2 | 32.0 | 48.7 |
| Zh→En | 6 | 6 | 22.2 | 62.4 |
| | | 2 | 21.6 | 62.3 |

**Model size**: The size of many machine translation models is optimized such that the computational hardware is used to its fullest extent. The two key restrictions are the total runtime of the training and the amount of memory of the provided hardware, typically a GPU. The number of network layers and nodes, the vocabulary size and the batch size are chosen in such a way that the memory of the GPU is fully used during model training. However, for all experiments in this section the GPU is used to run both a translation model and a language model. Thus during training, both models need to fit in the GPU memory, together with all hidden states needed for the computation of the backward pass. With the hardware available to the author it is impossible to train two models of common size simultaneously on the same GPU. Hence, throughout this section we reduce the model size of the translation model by four decoder layers compared to the 'base' transformer size. We only employ two decoder layers and all language models consist of six layers. The effect of this can be seen in Table 4.5. We observe that the smaller models lose 0.5-2.7 Bleu [%] compared to the base architecture with six decoder layers.

**Training Time** Reducing the model size alone is not sufficient to fit a fusion model on the GPU during training. To shrink the memory consumption further, we drastically reduce the batch size from around 5k (depending on the task) tokens to 350. However, the translation performance of the transformer architecture can become unstable if small batch sizes are used [Popel & Bojar 18], hence, we apply gradient accumulation and scale the effective batch size up to 9k tokens. Reducing the technical batch size affects the parallelization of the training and thus reduces the training speed. We report the training speed of the adjusted fusion models in Table 4.6. Instead of 12 minutes per epoch for a baseline model, a symbol-level fusion model requires around 35 minutes per epoch. This increase is caused by the need to compute the language model and the reduced technical batch size. For sequence-level normalization, training time increases by a

Table 4.6: Impact of language model fusion on the training time. For each model, the training parameters are set to make good use of the available GPU hardware. Training time is given per epoch.

| | TM #layers | | LM | batch size | | train time/epoch | |
|---|---|---|---|---|---|---|---|
| | enc | dec | | technical | effective | hh:min | factor |
| baseline | 6 | 6 | no | 4700 | 18k | 00:12 | 1.0 |
| | | 2 | | 4700 | 18k | 00:09 | 0.7 |
| local re-normalization | 6 | 2 | yes | 350 | 9k | 00:35 | 2.8 |
| global re-normalization | | | | 350 | 9k | 07:58 | 38.4 |

Table 4.7: Comparison of language model perplexities across all tasks. We compare a language model trained on the target side of the bilingual data with a language model that additionally uses monolingual target data. All results are presented on the development set.

| Task | data | trg | LM | TM | |
|---|---|---|---|---|---|
| | | # words | PPL (BPE) | BLEU [%] | TER [%] |
| En→It | bilingual | 4M | 53.4 | 26.9 | 55.0 |
| | monolingual | 46M | 34.4 | - | - |
| Ro→En | bilingual | 16M | 115.9 | 33.0 | 48.1 |
| | monolingual | 1G | 29.2 | - | - |
| De→En | bilingual | 118M | 84.0 | 32.0 | 48.7 |
| | monolingual | 1G | 39.7 | - | - |
| Zh→En | bilingual | 389M | 81.3 | 21.6 | 62.3 |
| | monolingual | 1G | 44.4 | - | - |

factor of almost 40. Since even the full training on our smallest task, English→Italian, takes 1-2 days of training time, this means that language model fusion experiments cannot be trained from scratch. Instead, we start from the fully converged baseline model for all language model fusion experiments.

### Language Model Fusion in Search

In this section, we report our findings on symbol-level and sequence-level normalization for language model fusion as described in Section 4.1. As a first step, we combine the language and translation model only during the search, meaning that we use the baseline cross-entropy training criterion defined in Equation 3.6 and the decision rules given by Equation 4.2, respectively Equation 4.4. The starting point for all four translation tasks is presented in Table 4.7, showing the data conditions as well as the individual translation and language model performances.

**Symbol-level normalization:** To decode with the symbol-level fusion model we use the Bayes' decision rule as described in Equation 4.2 and, in contrast to sequence-level normalization, the denominator needs to be computed during the search. As discussed previously, the full denominator can be computed and no approximation is needed. First, we investigate the scaling of the translation and language model. From the results in Table 4.8 we observe that the performance of the fusion model deteriorates if the translation model scale $\alpha$ and the language model scale $\beta$ have a ratio $\frac{\alpha}{\beta} \leq 1$. In other words, if the language model obtains a higher scale than the translation model the performance drops below 20.3 BLEU [%], in many cases even below 6.0 BLEU [%]. If the language model is weighted by a scale of $\beta = 10^{-4}$, i.e. it is almost completely

Table 4.8: English→Italian: Scales of the translation model ($\alpha$) and the language model ($\beta$) for symbol-level normalization. Models are trained independently and are combined only during the search. The baseline translation system obtains 26.9 Bleu [%] if no language model is used during the search and the language model has a perplexity of 34.4. Highlighted are the best-performing combination (26.9) and the combination used for further experiments (26.7). Bleu scores are reported on the development set.

| $\alpha$ \ $\beta$ | $10^{-4}$ | $10^{-3}$ | $10^{-2}$ | $10^{-1}$ | 0.5 | 1 | 5 | 10 |
|---|---|---|---|---|---|---|---|---|
| $10^{-3}$ | 25.1 | 5.6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.1 |
| $10^{-2}$ | 25.4 | 25.2 | 5.9 | 0.0 | 0.0 | 0.0 | 0.1 | 0.1 |
| $10^{-1}$ | 25.5 | 25.6 | 25.0 | 6.2 | 0.0 | 0.1 | 0.0 | 0.1 |
| 0.5 | 26.0 | 26.1 | 26.2 | 25.5 | 20.3 | 4.2 | 0.1 | 0.1 |
| 1.0 | **26.9** | 26.8 | 26.8 | **26.7** | 24.5 | 18.1 | 0.3 | 0.1 |
| 5.0 | 26.0 | 26.1 | 26.0 | 26.1 | 25.9 | 25.2 | 15.7 | 3.4 |
| 10.0 | 26.0 | 26.1 | 26.1 | 26.1 | 25.8 | 25.9 | 23.0 | 15.5 |

ignored during the search, the translation model is still affected by $\alpha$. Setting $\alpha < 1$ flattens the posterior distribution while $\alpha > 1$ sharpens it. Both approaches have a negative impact on translation performance, indicating that the sharpness of the distribution is well trained. The best results are obtained if the language model impact is reduced to a minimum $\beta = 10^{-4}$ and the scale of the translation model is set to the baseline value $\alpha = 1$. As expected, this yields the same performance as the baseline system with no external language model, reaching 26.9 Bleu [%]. In order to keep the impact of the language model meaningful, we argue that $\alpha = 1$ and $\beta = 10^{-1}$ is a better starting point for further experiments. This setup still yields 26.7 Bleu [%], i.e. a minor deterioration of 0.2 Bleu [%], while we still expect the language model to influence the fusion model.

Table 4.9: Romanian→English: Scales of the translation model ($\alpha$) and the language model ($\beta$) for sequence-level normalization. Models are trained independently and are combined only during the search. The baseline translation system obtains 33.0 Bleu [%] if no language model is used during the search. Highlighted is the best-performing combination. Bleu scores are reported on the development set.

| $\alpha$ \ $\beta$ | $10^{-4}$ | $10^{-3}$ | $10^{-2}$ | $10^{-1}$ | 0.5 | 1 | 5 | 10 |
|---|---|---|---|---|---|---|---|---|
| $10^{-3}$ | 32.5 | 15.8 | 0.3 | 0.1 | 0.1 | 0.2 | 0.2 | 0.1 |
| $10^{-2}$ | 31.7 | 32.5 | 15.8 | 0.2 | 0.1 | 0.1 | 0.2 | 0.1 |
| $10^{-1}$ | 31.7 | 31.8 | 32.5 | 16.1 | 1.0 | 0.6 | 0.2 | 0.3 |
| 0.5 | 32.0 | 32.0 | 32.2 | 33.1 | 25.7 | 7.1 | 0.5 | 0.4 |
| 1.0 | 33.0 | 33.0 | 33.0 | **33.2** | 30.8 | 21.9 | 1.1 | 0.5 |
| 5.0 | 31.9 | 31.9 | 31.9 | 32.0 | 32.3 | 31.8 | 18.5 | 5.3 |
| 10.0 | 31.8 | 31.8 | 31.8 | 31.8 | 31.9 | 32.0 | 27.9 | 18.5 |

To verify the scaling decision and confirm the previous observations, we consider the same experiment on the Romanian→English task. We observe the same trend in Table 4.9. Starting with a baseline performance of 33.0 Bleu [%] the performance of the fusion model drops by at least 6.0 Bleu [%] if the scale of the language model exceeds the scale of the translation model, i.e. $\beta > \alpha$. If $\beta$ is more than $10\times$ bigger than $\alpha$ the performance ranges between 0.1 and 7.1 Bleu [%]. Widening or sharpening of the translation model via $\alpha \neq 1$ leads to smaller, but significant, performance drops of around 1.0 Bleu [%]. Finally, $\alpha = 1$ and $\beta = 0.1$ is confirmed as a solid parameter choice, even yielding the best Bleu performance.

Table 4.10: Language model scale for sequence-level normalization, All results are reported on the development set and the best results of each column are highlighted in bold.

| $\gamma$ | En→It | | Ro→En | |
|---|---|---|---|---|
| | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| - | 26.9 | 54.9 | 33.0 | **48.1** |
| $10^{-4}$ | 26.9 | 54.9 | 33.0 | **48.1** |
| $10^{-3}$ | 26.8 | 54.9 | 33.0 | **48.1** |
| $10^{-2}$ | 26.9 | **55.0** | 33.0 | 48.2 |
| $10^{-1}$ | **27.2** | 55.2 | **33.6** | 48.3 |
| 0.5 | 23.9 | 62.3 | 30.4 | 55.5 |
| 1 | 13.4 | 96.8 | 1.0 | 191.8 |
| 10 | 0.0 | 132.7 | 0.1 | 198.6 |

**Sequence-level normalization**: In the next step we investigate fusion models that are normalized on the sequence level. As described in Equation 4.4, we omit the calculation of the denominator in the decision rule and simplify the language and translation model scale into a single weight $\gamma = \frac{\beta}{\alpha}$. Hence, bigger values of $\gamma$ mean that the language model is more influential during the search. The effects of different language model scales on English→Italian and Romanian→English are reported in Table 4.10. We observe that for $\gamma \leq 10^{-2}$ the impact of the language model is negligible since the fusion system performs almost identically to the baseline. A value of $\gamma = 10^{-1} = 0.1$ seems to be a sweet spot as we observe a small improvement of 0.3 Bleu [%] on the English→Italian task as well as 0.6 Bleu [%] for Romanian→English. However, in both cases we see a small degradation in the Ter score. If the impact of the language model grows further, we observe a rapid decline in translation quality. This is plausible; since the language model obtains no information about the source sentence, it should not be the leading factor for the translation decisions of the fusion model. We use $\gamma = 0.1$ for all further experiments on sequence-level normalized fusion models because it yields the best Bleu scores in our experiments and makes the language model as influential as possible without a major loss in Ter. Notably, a value of $0.1 = \gamma = \frac{\beta}{\alpha}$ corresponds to $\alpha = 1$ and $\beta = 0.1$, the optimal model scales we found for symbol-level normalization.

Table 4.11: Effect of language model fusion in search across two low-resource tasks. All results are reported on the development set.

| | external | En→It | | Ro→En | |
|---|---|---|---|---|---|
| | LM | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| baseline | no | 26.9 | 55.0 | 33.0 | 48.1 |
| symbol-level fusion | yes | 26.7 | 55.2 | 33.2 | 48.4 |
| sequence-level fusion | yes | 27.2 | 55.2 | 33.6 | 48.3 |

We apply the selected scales across all four language pairs and report the results in Table 4.11 for the two low-resource languages and in Table 4.12 for the high-resource tasks. Generally, Bleu is improved while we obtain a worse Ter score for all four experiments. This is a very common pattern when the length of the hypothesis changes since the two metrics incorporate the reference length differently. We verify whether the inclusion of a language model in search does affect the

Table 4.12: Effect of language model fusion in search across two medium- to high-resource tasks. All results are reported on the development set.

| | external | De→En | | Zh→En | |
|---|---|---|---|---|---|
| | LM | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| baseline | no | 32.0 | 48.7 | 21.6 | 62.3 |
| symbol-level fusion | yes | 32.8 | 48.7 | 21.9 | 62.9 |
| sequence-level fusion | yes | 33.1 | 49.0 | 22.2 | 63.3 |

Table 4.13: Length comparison between the reference and the hypothesis. A value smaller than 1.00 indicates a hypothesis that is too short while a value bigger than 1.00 denotes that the hypothesis is longer than the reference.

| | length ratio | | | |
|---|---|---|---|---|
| | En→It | Ro→En | De→En | Zh→En |
| reference | 1.00 | 1.00 | 1.00 | 1.00 |
| baseline | 1.02 | 0.97 | 0.97 | 0.94 |
| symbol-level fusion | 1.03 | 1.00 | 0.98 | 0.96 |
| sequence-level fusion | 1.03 | 1.00 | 0.99 | 0.97 |

length of the generated hypothesis in Table 4.13. It is clearly visible that language model fusion increases the hypothesis length by 1-3 percentage points and that sequence-level normalization tends to generate the longest sequences. It is important to note that the language model does not improve length modeling, as we can see from the English→Italian task. Despite the baseline system already generating a hypothesis of appropriate length, the length increases for language model fusion systems.

Overall, the strongest performance is obtained by sequence-level normalization; however, the improvements vary quite heavily across languages. The smallest improvement is observed on English→Italian (0.3 Bleu [%]) and the biggest on German→English (1.1 Bleu [%]). We conclude that language model fusion can help to improve the translation performance with respect to Bleu depending on the task. Furthermore, a clear increase in hypothesis length is visible, which affects the two automatic metrics differently.

The addition of a language model happens via log-linear model combination. This means that the fused model has significantly more parameters and could benefit from an ensemble learning effect. In the next step, we want to investigate whether the Bleu improvement for fusion models is

Table 4.14: Effect of the training data of the language model on the final fusion model. Language models are either trained only on the target side of the bilingual data (biling) or on all target data available (mono), i.e. monolingual target data and the target side of the bilingual data. Perplexity is reported on BPE level and all results are obtained on the development set.

| | external LM | | En→It | | | Ro→En | | |
|---|---|---|---|---|---|---|---|---|
| | used? | data | LM PPL | Bleu [%] | Ter [%] | LM PPL | Bleu [%] | Ter [%] |
| baseline | no | - | - | 26.9 | 55.0 | - | 33.0 | 48.1 |
| symbol-level fusion | yes | mono | 34.4 | 26.7 | 55.2 | 29.2 | 33.2 | 48.4 |
| | | biling | 53.4 | 26.8 | 55.2 | 115.9 | 32.8 | 48.7 |
| sequence-level fusion | yes | mono | 34.4 | 27.2 | 55.2 | 29.2 | 33.6 | 48.3 |
| | | biling | 53.4 | 26.8 | 55.5 | 115.9 | 32.7 | 49.1 |

Table 4.15: German→English: Effect of the training data of the language model on the final fusion model. Language models are either trained only on the target side of the bilingual data (biling) or on all target data available (mono), i.e. monolingual target data and the target side of the bilingual data. Perplexity is reported on BPE level and all results are obtained on the development set.

| | external LM | | De→En | | |
|---|---|---|---|---|---|
| | used? | data | LM PPL | BLEU [%] | TER [%] |
| baseline | no | - | - | 32.0 | 48.7 |
| symbol-level fusion | yes | mono | 39.7 | 32.8 | 48.7 |
| | | biling | 84.0 | 32.3 | 49.3 |
| sequence-level fusion | yes | mono | 39.7 | 33.1 | 49.0 |
| | | biling | 84.0 | 32.4 | 49.9 |

Table 4.16: Domain analysis on global-normalized fusion models. Data originates from popular scientific talks (science), news articles (news) and parliamentary speeches (parl.). The test set printed in bold is the official test set of the corresponding task.

| task | domain of data | | | $\text{LM}_{\text{biling}}$ | $\text{LM}_{\text{mono}}$ | TM | TM × $\text{LM}_{\text{mono}}$ |
|---|---|---|---|---|---|---|---|
| | bilingual | monolingual | test | PPL | PPL | BLEU | BLEU |
| En→It | science | news | **science** | 53.4 | 34.4 | 26.9 | 27.2 |
| | | | news | 94.6 | 22.6 | 20.5 | 21.7 |
| De→En | parl., other | news | parl. | 24.0 | 25.9 | 33.1 | 33.5 |
| | | | **news** | 84.0 | 39.7 | 32.0 | 33.1 |

an effect of the additional monolingual training data or whether the log-linear model combination itself is helpful. In Table 4.14 we compare the impact of two language models during the search: one is trained only on the target side of the bilingual data while the other also uses external monolingual data. First, we observe that the language models that include additional data obtain better perplexity values, thus we expect them to be more beneficial in a fusion model. Secondly, we see that language models which make use of the full data available outperform their data-restricted counterparts. Since the performance differences are very small in both English→Italian and Romanian→English, we perform the same experiment on the German→English task, which showed the biggest language model impact (see Table 4.12). Both observations are consistent with the German→English task (see Table 4.15).

**Domain of data** An important aspect of machine learning is the domain of the data involved. Bilingual data usually can only be obtained from special sources, which provide a sentence-by-sentence translation of a reasonable amount of textual data. In contrast to this, every English text written is a source of monolingual English data that can be collected. Due to this, both the number as well as the variety of text domains are commonly limited for bilingual data. In this work, we test our methods on shared translation tasks provided by the IWSLT and WMT, where the test data stems from the science and news domains respectively. Depending on the language pair involved, this matches the domain of either the bilingual or the monolingual data. In Table 4.16 we give an overview of the datasets and domains involved. Furthermore, we compare the perplexity of a language model that was trained on the target side of the bilingual data with a language model that also uses the monolingual data. We can assume that the model with restricted training data approximates the language modeling capabilities of a translation model since they are similar in architecture and equal in target-side information. For both English→Italian and German→English we first consider a test set that shares the domain with the bilingual training data. In these cases the perplexity of both language models is comparatively close, although in the case of English→Italian there is a gap of roughly 20 Ppl. Keep in mind that this task consists of only 230k bilingual sentences, meaning that there are roughly 270 times more Italian words in the monolingual data than in the bilingual. For both languages, we observe only a small improvement of roughly +0.4 Bleu [%] for the fusion model that contains the language model trained on the monolingual news data. However, if we consider a test set from the news domain, which matches the monolingual data, the gap in language model perplexities increases drastically. This improvement carries over to the performance of the fusion model, where we report an improvement of roughly 1.1 Bleu [%] across both language pairs. Notably, these findings are consistent across both language pairs with very different bilingual data sizes. We conclude that the domain of all datasets involved in the training and evaluation of the models is a crucial factor that can shift the effects of language model integration from barely noticeable to quite significant. In the following, we will continue to report results on the standard development and test sets of the respective tasks to maintain comparability with the wider research community. We address the impacts of data domains again in Section 4.4.5.

So far we have found that monolingual text data contains information that can in principle be helpful even to high-resource tasks such as Chinese→English, meaning that even a bilingual corpus of 17M sentence pairs is not sufficient to fully learn the structure of the target language. Furthermore, we observe that language model fusion in search does improve Bleu scores consistently but the effect varies heavily across tasks. Besides the amount of bilingual training data, the domain of all datasets involved in training and evaluation is a leading factor in determining whether language model fusion will help a certain task.

### Language Model Fusion in Training

So far we considered the fusion of a translation and a language model at decoding time. Next, we investigate models where the translation model is trained given the existing language model. Since the training of the fusion model is very slow (see Table 4.6), training the machine translation model from scratch using a joint model loss is not an option. Hence, we use the fully trained language and machine translation models from the previous section and fine-tune the fusion model. Note that only the parameters of the translation model are optimized during the fine-tuning process.

Table 4.17: Applying LMs trained on different datasets for symbol-level normalized fusion models. All results are reported on the development set.

| LM used in | | En→It | | Ro→En | | De→En | |
|---|---|---|---|---|---|---|---|
| training ($p_{\text{t-LM}}$) | search ($p_{\text{s-LM}}$) | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| none | none | 26.9 | 55.0 | 33.0 | 48.1 | 32.0 | 48.7 |
| biling | biling | 26.9 | 55.2 | 33.0 | 48.4 | 32.4 | 49.1 |
| | mono | 27.0 | 55.1 | 33.4 | 47.8 | 32.8 | 48.3 |
| mono | biling | 26.9 | 55.0 | 33.0 | 48.3 | 32.4 | 49.0 |
| | mono | 27.0 | 55.0 | 33.5 | 47.9 | 32.8 | 48.7 |

**Symbol-level normalization:** First we train fusion models with symbol-level normalization according to the training criterion described in Equation 4.1, where normalization of the fusion model is applied for every target position $i$. We use the optimal parameters for the translation model scale $\alpha = 1$ and the language model scale $\beta = 0.1$ that we obtained from the local fusion experiments in search (see Table 4.8 and Table 4.9). Search is performed using the decision rule defined in Equation 4.2. Note that we distinguish between the language model used in training $p_{\text{t-LM}}$ and that used in search $p_{\text{s-LM}}$. We consider a language model trained either on the target side of the bilingual data $p_{\text{biling}}$ or on all target data available $p_{\text{mono}}$, specifically the target side of the bilingual data as well as the monolingual data. From the results reported in Table 4.17 we observe that the overall impact of language model fusion is very limited. Models trained on English→Italian are not affected by the inclusion of a language model. We see small improvements of around 0.5 Bleu [%] and 0.3 Ter[%] for Romanian→English if the full data language model $p_{\text{mono}}$ is used. The results on German→English confirm that the language model used in search $p_{\text{s-LM}}$ is more important than the one used in training $p_{\text{t-LM}}$. Here we report the biggest improvement of 0.8 Bleu [%] when $p_{\text{mono}}$ is used for language model fusion independently of the language model used in training. Note that the improvement does not carry over to Ter. Overall, the inclusion of the data-restricted language model in search $p_{\text{biling}} = p_{\text{s-LM}}$ does not lead to significant improvements.

Table 4.18: Comparison of denominator approximation strategies during training for sequence-level normalization in fusion models. All results are reported on the development set.

| re-norm. | decoder approx. | En→It Bleu [%] | En→It Ter [%] | Ro→En Bleu [%] | Ro→En Ter [%] | De→En Bleu [%] | De→En Ter [%] |
|---|---|---|---|---|---|---|---|
| none | - | 26.9 | 55.0 | 33.0 | 48.1 | 32.0 | 48.7 |
| sequence-level | 12-best | 27.2 | 55.2 | 33.6 | 48.3 | 33.1 | 49.0 |
| | trellis | 27.2 | 55.2 | 33.6 | 48.2 | 33.1 | 49.0 |

**Sequence-level normalization**: Equation 4.5 defines the training criterion used for fusion models with sequence-level normalization. During the search we apply the decision rule as defined in Equation 4.4 where the denominator is dropped. We consider two approximations of the denominator, as portrayed in Section 4.1, either via an $n$-best list or a trellis with limited context assumption on the target side. We choose a translation model scale $\alpha = 1$ and a language model scale $\beta = 0.1$ during training, corresponding to the best results in search (Table 4.10). We set the beam size to $n = 12$ and generate a 12-best list, meaning that the twelve hypotheses with the best translation model score are used in the denominator approximation; however, we ensure that the reference is always included. For the trellis approximation, we use a merging history of $k = 1$ for our initial experiments. From the results in Table 4.18 we observe that both approximation schemes for the denominator result in almost identical results.

Table 4.19: Comparing the effect of using a language model in a fusion model with sequence-level normalization. All results are reported on the development set.

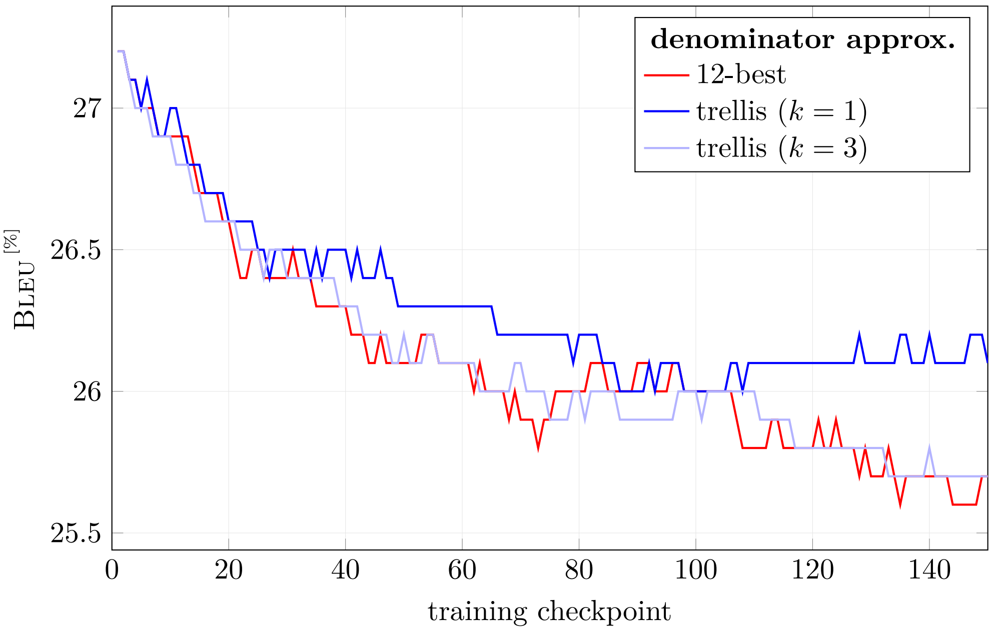
| LM used in search | LM used in train | En→It Bleu [%] | En→It Ter [%] | Ro→En Bleu [%] | Ro→En Ter [%] | De→En Bleu [%] | De→En Ter [%] |
|---|---|---|---|---|---|---|---|
| none | none | 26.9 | 55.0 | 33.0 | 48.1 | 32.0 | 48.7 |
| biling | none | 26.8 | 55.5 | 32.7 | 49.1 | 32.4 | 49.9 |
| | biling | 26.8 | 55.5 | 33.0 | 48.4 | 32.4 | 50.0 |
| | mono | 26.8 | 55.5 | 32.6 | 49.2 | 32.4 | 50.0 |
| mono | none | 27.2 | 55.2 | 33.6 | 48.3 | 33.1 | 49.0 |
| | biling | 27.2 | 55.3 | 33.2 | 48.6 | 33.1 | 49.0 |
| | mono | 27.2 | 55.2 | 33.6 | 48.3 | 33.1 | 49.0 |

Next, we consider the effect of applying different language models $p_{\text{t-LM}}, p_{\text{s-LM}}$ during training and search. From the results in Table 4.19 we conclude that the selection of the language model $p_{\text{t-LM}}$ used in training does not affect the model performance.

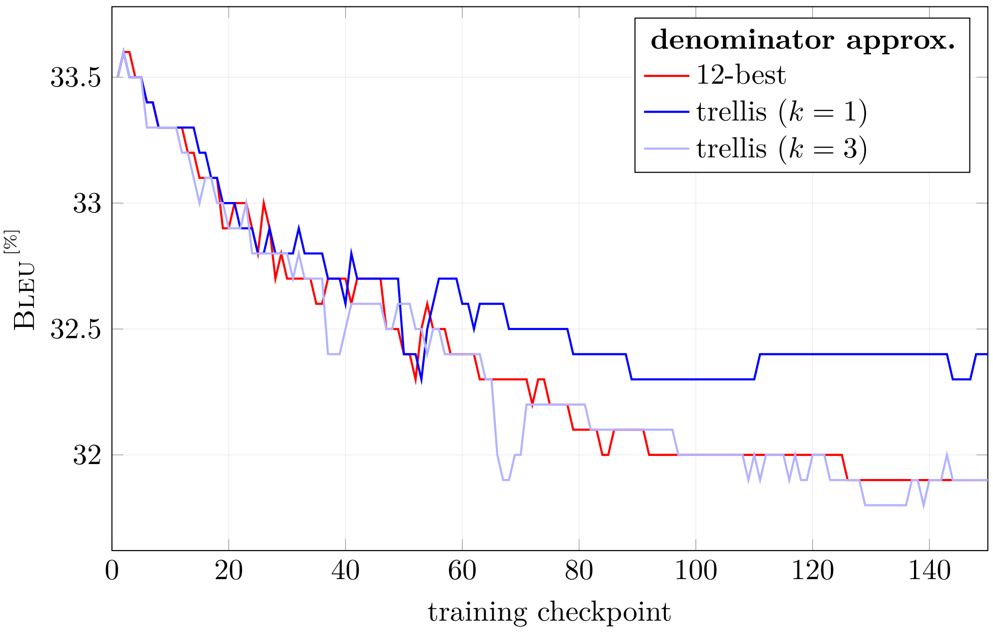
To investigate this, we consider the development of the Bleu score during the training of a fusion model with sequence-level normalization. From Figure 4.8 we observe that the Bleu score

Figure 4.8: Bleu score of sequence-level normalized fusion models trained with different denominator approximations.

(a) English→Italian



(b) Romanian→English



slowly but relatively consistently drops throughout the training. This effect is consistent across the different denominator approximations using either an $n$-best list or a trellis. Furthermore, the context length $k$ of the trellis has only a minor impact on the training behavior. Note that the fusion model does not diverge with respect to the development Bleu score but instead seems to converge to a stable score roughly 1-1.5 Bleu $^{[\%]}$ below the starting point.

A reduction in Bleu score over the course of the training is a serious problem since it shows that the model performs worse on the intended task the more it is optimized. This raises the question of whether the optimization is failing or whether there is a mismatch between the training criterion and the final objective. We consider the training criterion defined in Equation 4.5

$$\begin{aligned} F_{\text{global}} &= \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log \frac{\prod_{i=1}^{I} p_{\text{TM}}(e_i|e_0^{i-1}, f_1^J)^\alpha p_{\text{t-LM}}(e_i|e_0^{i-1})^\beta}{\sum_{\tilde{I}, \tilde{e}_0^{\tilde{I}}} \prod_{i=1}^{\tilde{I}} p_{\text{TM}}(\tilde{e}_i|\tilde{e}_0^{i-1}, f_1^J)^\alpha p_{\text{t-LM}}(\tilde{e}_i|\tilde{e}_0^{i-1})^\beta} \\ &= \underbrace{\sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log \left( \prod_{i=1}^{I} p_{\text{TM}}(e_i|e_0^{i-1}, f_1^J)^\alpha p_{\text{t-LM}}(e_i|e_0^{i-1})^\beta \right)}_{Q} \\ &\quad \underbrace{- \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log \left( \sum_{\tilde{I}, \tilde{e}_0^{\tilde{I}}} \prod_{i=1}^{\tilde{I}} p_{\text{TM}}(\tilde{e}_i|\tilde{e}_0^{i-1}, f_1^J)^\alpha p_{\text{t-LM}}(\tilde{e}_i|\tilde{e}_0^{i-1})^\beta \right)}_{Z} \end{aligned} \tag{4.14}$$

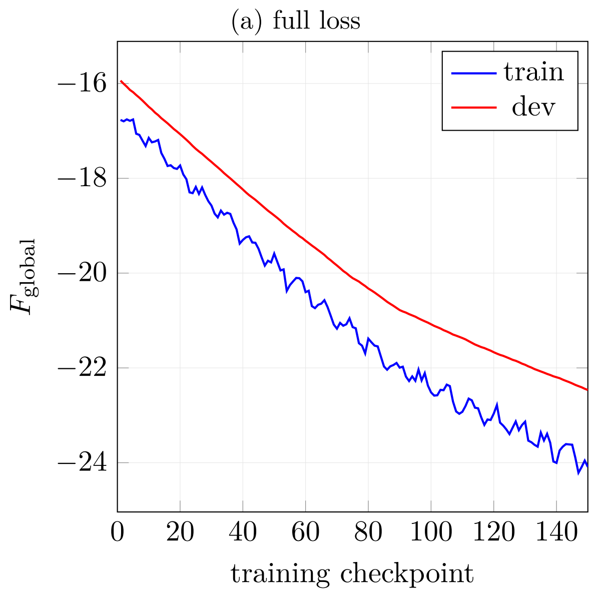
and plot it in Figure 4.9 together with the logarithm of its numerator $Q$ and the negative logarithm of its denominator $Z$. First, we observe that the overall loss is reduced over the course of the training, both on the training and the development data, meaning that the optimization is working as intended. However, looking into the two parts of the loss function, we observe that the improvements originate solely from a reduction of $Z$ while there is even a small increase in $Q$. This means that the probability mass assigned to the reference translation decreases slightly throughout the training. At the same time, probability mass is shifted away from the sequences in the denominator, which in the case of the $n$-best list approximation, are often closely related to the reference. It seems that both the correct translation as well as promising alternatives are discouraged by the sequence-level re-normalization, resulting in an overall worse translation performance. We conclude that the optimization of the fusion model with sequence-level normalization works but there is either a mismatch between the training criterion and the Bleu metric or the approximation of the denominator is still too weak.

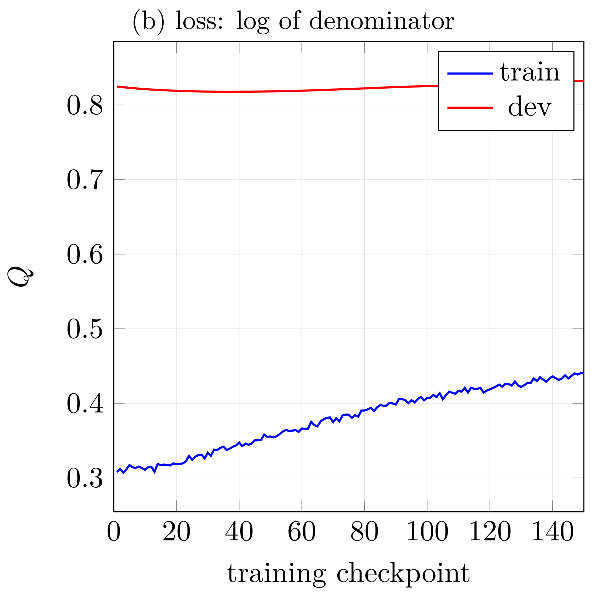
### Automatic Speech Recognition

Sequence-level normalization for fusion models shows no improvements and problematic training behavior in the case of machine translation. However, these methods have been are successfully applied to the task of automatic speech recognition (ASR) by a research team involving the author of this work [Wynands & Michel$^+$ 22]. Results on the task of LibriSpeech are reported in Table 4.20. All results are taken from the ICASSP publication [Wynands & Michel$^+$ 22] and the data as well as the setup used are described there. The work is an extension of [Michel & Schlüter$^+$ 20], which showed that language model integration in training can be helpful for ASR systems.

On the task of automatic speech recognition we observe an improvement in WER on the LibriSpeech task. The trellis and the $n$-best list approximation of the denominator perform equally well, indicating that a bigger search space for the denominator approximation is not needed.

Figure 4.9: Sequence-level normalization training for fusion models where the denominator is computed with the trellis approximation. We show (a) the full loss $F_{\text{global}}$ is split into two components: (b) the enumerator $Q$ and (c) the denominator $Z$ (see Equation 4.14). We observe that the loss reduction in training is obtained entirely through a drop in the denominator; the enumerator of the loss increases slightly over the course of the training. Results are reported on the Romanian→English task.





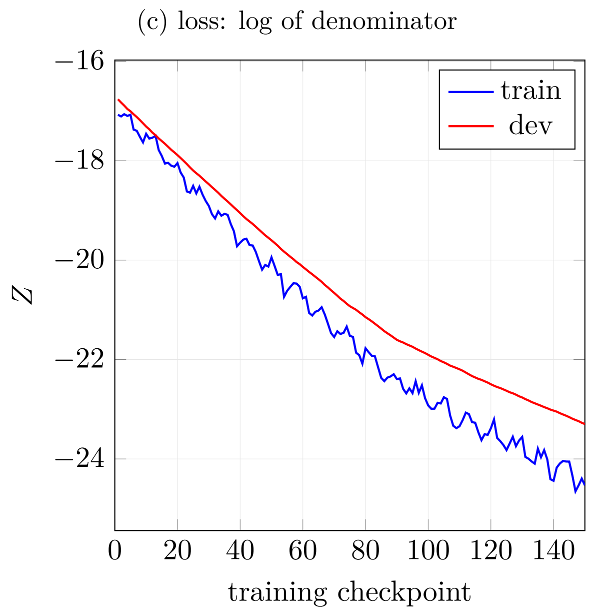


Table 4.20: Results of sequence-level fusion models on the LibriSpeech task of automatic speech recognition. All results are taken from [Wynands & Michel$^+$ 22].

| re-norm | decoder approx. | dev WER[%] clean | dev WER[%] other | test WER[%] clean | test WER[%] other |
|---|---|---|---|---|---|
| none | - | 2.4 | 6.7 | 2.7 | 7.0 |
| sequence-level | n-best | 2.1 | 5.8 | 2.3 | 6.5 |
| | trellis | 2.1 | 5.8 | 2.3 | 6.5 |

### 4.4.3 Pre-Training and Multi-Task Learning

Another way to use monolingual data is via pre-training and multi-task learning. First, we consider the case of pre-training as described in Section 4.2 before switching to multi-task learning (see Section 4.2.2). Pre-training approaches add an additional training step prior to the translation training, which causes additional computational overhead. However, the speed of the main training is not reduced, hence there is no need for model or batch size reduction, as described in the fusion experiments (see Section 4.4.2).

#### Experimental Setup: Monolingual Source Data

Most approaches that employ monolingual data to increase machine translation performance use only target data. However, the approaches in this section can make use of either source or target data, depending on which component of the network is pre-trained. For all tasks in this section, we use the news crawl data[12] released for the WMT conference [Barrault & Bojar$^+$ 19]. This gives us a rich corpus for most languages with 1 billion words or more. The only two exceptions are Chinese, for which the data does not provide a consistent word splitting, and Romanian, for which relatively few monolingual sentences are available. The detailed data statistics are reported in Appendix A.1.

#### Pre-Training

Pre-training describes one or more additional training runs that provide a suitable initialization point from which the main training starts. Consequently, all translation models in this section are trained with the baseline training criterion from Equation 3.6 and use the baseline decision rule as formulated in Equation 3.11b. The details of the training objectives for the pre-training are described in Section 4.2.1.

All language models in this section are six layers deep and match the perplexity of the language models for fusion models (see Section 4.4.2). In order to facilitate compatibility between the hidden states of the encoder and the decoder, the final system uses a shared source-target vocabulary with the same number of BPE merging operations as the baseline. Pre-training is performed using the Adam optimizer starting with a learning rate of $\lambda = 0.0001$ and newbob learning rate schedule, scaling it by a factor of 0.9 if the loss on the development set fails to improve for 4 consecutive checkpoints. The checkpoint with the best perplexity on the development set is selected as the starting point for the main training. For the main training the momentum terms of the Adam optimizer are reset and the learning rate of the baseline training is used.

For all pre-train experiments that apply input masking, we use the same masking scheme and default parameters of the BERT model [Devlin & Chang$^+$ 19], namely we distort 15% of the input

[12]`https://data.statmt.org/news-crawl/`

and each distortion is applied with the probability

$$\begin{aligned} \rho_{\text{mask}} &= 80\% \\ \rho_{\text{keep}} &= 10\% \\ \rho_{\text{random}} &= 10\% \end{aligned}$$

where the decision is stochastically independent for each token pair. Unlike in the original BERT paper, we do not have document-level data and omit the next sentence prediction (NSP) objective. We believe that this does not hurt the overall performance since several studies showed that the NSP objective is not helpful for most downstream tasks [Liu & Ott$^+$ 19, Joshi & Chen$^+$ 20].

Table 4.21: Effect of using monolingual source data for different strategies to pre-train the encoder on two low-resource tasks. All results are reported on the development set.

| additional src | additional | En→It | | Ro→En | |
|---|---|---|---|---|---|
| data | loss | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| no | none | 28.1 | 54.0 | 35.7 | 45.3 |
| yes | lm | 29.4 | 52.6 | 36.7 | 44.6 |
| | cloze | 29.4 | 52.2 | 36.8 | 44.3 |

Table 4.22: Effect of using monolingual source data for different strategies to pre-train the encoder on two high-resource tasks. All results are reported on the development set.

| additional data | additional | De→En | | Zh→En | |
|---|---|---|---|---|---|
| src | loss | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| no | none | 32.7 | 48.6 | 22.2 | 62.4 |
| yes | lm | 33.1 | 48.1 | 22.3 | 62.2 |
| | cloze | 33.1 | 48.0 | 22.5 | 61.6 |

**Monolingual source data:** First we consider pre-training approaches that rely solely on source data in order to create better source sentence representations within the encoder. The training criterion for the encoder pre-training tasks is either the cross-entropy for language modeling (Equation 4.7) or a cloze loss as used in the BERT model (Equation 4.9). The results are described in Table 4.21. We observe that English→Italian and Romanian→English benefit heavily from either pre-training approach. For English→Italian we see encoder pre-training improves the performance by 1.3 Bleu [%] and an equal amount of Ter [%] while Romanian→English improves by 1 point across both metrics. However, these improvements do not carry over to high-resource tasks. On German→English we report a small improvement of around +0.6 Bleu [%] and Ter [%], while there is no improvement on Chinese→English. Comparing the impact of the language model and the cloze loss, we find extremely similar performance, with a minor lead by the cloze loss on each task. Since there is not a single dataset or metric in which the language model loss yields a better result, we consider the cloze loss for all further experiments that use encoder pre-training.

We conclude that source data is a valuable resource that can benefit low-resource machine translation systems. This is an important finding since traditionally only monolingual target data has been incorporated in a machine translation model. Notably, the more bilingual data available for a task, the less benefit is obtained from additional source data.

**Monolingual target data:** Similar to the encoder pre-training, we perform pre-training of the decoder using monolingual target data. We either pre-train the decoder as a language model, ignoring its cross-attention layer, or, to simulate the existence of a source sentence, we use a pseudo-encoder as described (see Section 4.2.1 for details). In the second case, we initialize the

Table 4.23: Effect of using monolingual target data for different pre-training strategies on two low-resource tasks. Depending on the approach, different parts of the model are pre-trained. All results are reported on the development set.

| additional trg | pre-trained | En→It | | Ro→En | |
|---|---|---|---|---|---|
| data | component | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| no | none | 28.1 | 54.0 | 35.7 | 45.3 |
| yes | dec | 28.9 | 53.2 | 35.9 | 45.3 |
| | enc+dec | 28.9 | 53.0 | 37.1 | 44.5 |

Table 4.24: Effect of using monolingual target data for different pre-training strategies on two high-resource tasks. Depending on the approach, different parts of the model are pre-trained. All results are reported on the development set.

| additional trg | pre-trained | De→En | | Zh→En | |
|---|---|---|---|---|---|
| data | component | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| no | - | 32.7 | 48.6 | 22.2 | 62.4 |
| yes | dec | 33.0 | 48.5 | 22.4 | 62.4 |
| | enc+dec | 32.7 | 48.6 | 21.9 | 63.0 |

full machine translation model with the pre-trained parameters, including the encoder. As we can see from Table 4.23, both pre-training approaches yield improvements, but their performance varies across the tasks. For English→Italian pre-training, with target data yields an improvement of 0.8 Bleu [%] and 0.8-1.0 Ter [%]. On the Romanian→English task, we observe no significant improvement in Bleu and Ter when pre-training only the decoder. However, the full model pre-training via a pseudo-encoder improves the system by 1.4 Bleu [%] and 0.8 Ter [%]. For the larger tasks shown in Table 4.24, we observe no consistent improvement, with all results in a range of 0.3 Bleu around the baseline. We conclude that pre-training with target-side data is an effective technique for low-resource tasks but loses value as more bilingual data becomes available for the main training. This finding is consistent with the behavior for source data pre-training.

**Analysis of improvements:** To verify whether the improvements on English→Italian and Romanian→English for any of the proposed pre-training strategies are an effect of the additional data or the additional loss, we perform pre-training using only the bilingual training data. For these experiments we pre-train the models involved using either only the source or only the target side of the bilingual data. This yields a comparatively small monolingual corpus. From the results in Table 4.25, we observe that roughly half of the pre-training improvement stems from the use of external data if the additional loss is applied directly to the encoder. This is a surprising result that indicates that the training signal in the encoder is not strong enough during training. If the pre-training loss is applied to the decoder, the benefit of monolingual data is less notable. On the English→Italian task we observe around 0.1-0.3 improvement across Ter and Bleu, which is commonly not considered significant. For Romanian→English we report an increase by 0.4-0.5 Bleu [%] using the target side of the bilingual data and 1.2-1.4 Bleu [%] for the full data. We conclude that the encoder benefits from a stronger training signal as well as the additional data, while the decoder mainly improves based on the new target data.

Table 4.25: Pre-training strategies compared on two different data conditions.

| pre-trained component | pre-train loss | extra data src | extra data trg | En→It Bleu [%] | En→It Ter [%] | Ro→En Bleu [%] | Ro→En Ter [%] |
|---|---|---|---|---|---|---|---|
| none | none | | | 28.1 | 54.0 | 35.7 | 45.3 |
| enc | $\text{lm}_{\text{src}}$ | | | 28.8 | 53.0 | 36.2 | 45.0 |
| | | yes | | 29.4 | 52.6 | 36.7 | 44.6 |
| | cloze | | | 28.9 | 52.8 | 35.8 | 45.6 |
| | | yes | | 29.4 | 52.5 | 36.8 | 44.3 |
| dec | $\text{lm}_{\text{trg}}$ | | | 28.2 | 53.7 | 36.1 | 45.4 |
| | | | yes | 28.9 | 53.2 | 35.9 | 45.3 |
| enc+dec | | | | 28.4 | 53.7 | 36.2 | 44.8 |
| | | | yes | 28.9 | 53.0 | 37.1 | 44.5 |

Table 4.26: Findings on the effect of different pre-training strategies using monolingual source or target data. Depending on the type of data, different parts of the model are pre-trained using either a cloze or language model loss. All results are reported on the test set.

| mono data src | mono data trg | pre-trained component | additional loss | En→It `tst2010` Bleu [%] | En→It `tst2010` Ter [%] | Ro→En `newstest2016` Bleu [%] | Ro→En `newstest2016` Ter [%] |
|---|---|---|---|---|---|---|---|
| | | none | none | 29.6 | 52.0 | 34.1 | 47.4 |
| yes | | enc | cloze | 31.0 | 50.6 | 35.2 | 46.5 |
| | yes | enc + dec | $\text{lm}_{\text{trg}}$ | 31.1 | 50.8 | 35.4 | 46.5 |
| yes | yes | enc + dec | $\text{lm}_{\text{src}}$, $\text{lm}_{\text{trg}}$ | 31.3 | 50.3 | 35.3 | 46.3 |

Finally, we report the results of the combination of the best-performing encoder and decoder pre-train methods on the unseen test sets of all four tasks in Tables 4.26 and Table 4.27. The results highlight the central findings of this section:

1. Monolingual data in general and pre-training in particular are beneficial.
2. Source data and the pre-training of the encoder are helpful across all tasks, but the impact is inverse to the amount of bilingual data. This is an effect of both stronger training feedback as well as additional data.
3. Pre-training the decoder is very helpful for low-resource tasks and does not affect high-resource tasks. Any improvements on the decoder side are due to its exposure to additional data.

The gains from monolingual source data in the encoder are particularly interesting for the research community since most research on monolingual data focuses on the use of target data.

### Multi-Task Learning

For multi-task training, we optimize the translation system on two different tasks during the same training run. As in the case of pre-training, we use monolingual data to apply a language model or cloze loss as described in Equations 4.11–4.13 to the translation model. For pre-training, a sub-model is trained in a separate step and the resulting parameters are transferred to the initialization of the translation model. In contrast to this, in multi-task training the model is

Table 4.27: Findings on the effect of different pre-training strategies using monolingual source or target data. Depending on the type of data, different parts of the model are pre-trained using either a cloze or language model loss. All results are reported on the test set.

| mono data src | mono data trg | pre-trained component | additional loss | De→En newstest2017 BLEU [%] | De→En newstest2017 TER [%] | Zh→En newstest2017 BLEU [%] | Zh→En newstest2017 TER [%] |
|---|---|---|---|---|---|---|---|
| | | none | none | 33.5 | 48.2 | 23.0 | 61.9 |
| yes | | enc | cloze | 34.2 | 47.4 | 23.5 | 61.0 |
| | yes | enc + dec | $\text{lm}_{\text{trg}}$ | 33.8 | 48.0 | 22.7 | 62.3 |
| yes | yes | enc + dec | $\text{lm}_{\text{src}}$, $\text{lm}_{\text{trg}}$ | 34.1 | 47.5 | 23.3 | 61.1 |

optimized to perform both tasks simultaneously. This prevents the model from suffering from catastrophic forgetting, which describes a process where the pre-trained model parameters are shifted in such a way that they no longer constitute a solution for the pre-trained task.

Table 4.28: Multi-task learning strategies.

| mono data src | mono data trg | affected component | additional loss | En→It BLEU [%] | En→It TER [%] | Ro→En BLEU [%] | Ro→En TER [%] |
|---|---|---|---|---|---|---|---|
| | | none | none | 28.1 | 54.0 | 35.7 | 45.3 |
| yes | | enc | $\text{lm}_{\text{src}}$ | 28.3 | 53.5 | 35.5 | 45.7 |
| yes | | | cloze | 28.2 | 53.7 | 35.6 | 45.8 |
| | yes | dec | $\text{lm}_{\text{trg}}$ | 29.0 | 52.8 | 36.3 | 45.1 |
| yes | yes | enc+dec | cloze,$\text{lm}_{\text{trg}}$ | 28.1 | 53.9 | 36.0 | 45.2 |

Table 4.29: Multi-task learning strategies.

| mono data src | mono data trg | affected component | additional loss | De→En BLEU [%] | De→En TER [%] | Zh→En BLEU [%] | Zh→En TER [%] |
|---|---|---|---|---|---|---|---|
| | | none | none | 32.7 | 48.6 | 22.2 | 62.4 |
| yes | | enc | $\text{lm}_{\text{src}}$ | 32.2 | 49.2 | 22.1 | 62.7 |
| yes | | | cloze | 32.0 | 49.5 | 22.0 | 63.3 |
| | yes | dec | $\text{lm}_{\text{trg}}$ | 33.1 | 48.4 | 22.6 | 61.7 |
| yes | yes | enc+dec | cloze,$\text{lm}_{\text{trg}}$ | 32.2 | 49.3 | 22.3 | 63.0 |

The results of multi-task training are shown in Tables 4.28 and 4.29 for the low- and high-resource tasks respectively. We observe that an additional loss on the encoder does not improve the translation performance; while we see insignificant improvements for English→Italian, there are performance losses of comparable size for both Romanian→English and Chinese→English. The German→English system even worsens by 0.7 BLEU [%] and 1.1 TER[%]. Overall, the use of source data with an additional loss has either no or a negative impact on the system. Target-side data provides a consistent improvement of 0.4–0.9 BLEU [%] across all languages, with both low-resource settings gaining bigger improvements than German→English and Chinese→English.

We conclude that multi-task learning is not helpful to train the encoder using source data. However, the use of target data in encoder loss can be beneficial, especially if bilingual training data is scarce.

### 4.4.4 Back-Translation

Back-translation is the most common way to use monolingual data in machine translation systems. In this work, we only consider the standard back-translation introduced to neural machine translation by [Sennrich & Haddow$^+$ 16b] and described in Section 4.3.

Back-translation is a computationally costly process that involves the training of an additional translation system as well as the generation of synthetic source data and its processing during the training of the final system. While the training of an additional machine translation model can be considered a constant computational cost, the question of how much target data should be translated remains important. More synthetic data costs initial translation time and increases the training time for a full epoch on the final system. Furthermore, the amount of synthetic and original training data needs to be balanced to avoid overfitting to the synthetic data.

Table 4.30: Back-translation with different data ratios. The amount of original bilingual data is the same for all experiments and the amount of synthetic data is selected according to the data ratio. All results are reported on the development set.

| data ratio | En→It | | Ro→En | |
|---|---|---|---|---|
| human:synthetic | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| none (baseline) | 28.1 | 54.0 | 35.7 | 45.3 |
| 1:0.5 | 28.7 | 53.3 | 36.5 | 44.8 |
| 1:1 | 28.8 | 53.5 | 37.0 | 44.2 |
| 1:1.5 | 29.2 | 53.2 | 37.4 | 44.3 |
| 1:2 | 29.6 | 52.9 | 37.3 | 44.4 |
| 1:5 | 29.8 | 52.4 | 37.4 | 44.0 |
| 1:10 | 29.9 | 52.5 | 37.9 | 43.9 |

We test different amounts of training data for the two low-resource tasks English→Italian and Romanian→English and report the results in Table 4.30. For all experiments, we select a certain amount of synthetic data and then oversample the original bilingual data in such a way that both data conditions are evenly represented. For example, the original-to-synthetic data ratio of 1:2 for the English→Italian task in Table 4.30 represents a scenario where we use all 230k bilingual sentence pairs and add an additional 460k sentence pairs of monolingual data. Oversampling the original data by a factor of two results in a total of $4 \times 230\text{k} = 920\text{k}$ sentence pairs. We observe that more synthetic data generally helps more, but beyond a ratio of 1:2 we obtain diminishing returns. Overall, both systems improve significantly, with +1.8 BLEU [%] for the English→Italian setup and +2.2 BLEU [%] in the case of Romanian→English.

Table 4.31: Impact of back-translation on two high-resource tasks. All results are reported on the development set.

| system | De→En | | Zh→En | |
|---|---|---|---|---|
| | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| baseline | 32.7 | 48.6 | 22.2 | 62.4 |
| + back-translation | 35.1 | 45.8 | 23.0 | 61.3 |

Since back-translation is not the focus of this thesis, we conduct the corresponding experiments on the two high-resource language pairs with the greatest amount of synthetic data feasible. In the case of German→English this is a ratio of 1:5 and for Chinese→English we use a ratio of 1:1, totaling 46M, respectively 34M, sentence pairs.

Table 4.32: English→Italian: Overview over all methods using monolingual data.

| monolingual data usage | | mono data | | dev | | tst2010 | |
|---|---|---|---|---|---|---|---|
| | | src | trg | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| none (baseline) | | - | - | 28.1 | 54.0 | 29.6 | 52.0 |
| back-translation | | - | 0.1G | 29.9 | 52.5 | 31.4 | 50.5 |
| new loss | pre-training (enc,dec) | 2.5G | 1.2G | 29.6 | 52.5 | 31.3 | 50.3 |
| | multi-task (dec) | - | 1.2G | 29.0 | 52.8 | 30.5 | 50.8 |
| lm fusion | sequence normalization | - | 1.2G | 28.0 | 54.6 | 30.2 | 52.1 |
| | symbol normalization | - | 1.2G | 27.7 | 54.8 | 29.9 | 52.2 |

These improvements are consistent with the reports of other groups on the impact of back-translation [Sennrich & Haddow+ 16b].

### 4.4.5 Comparison of the Presented Methods

So far we have investigated three major approaches individually, namely language model fusion, pre-training and back-translation. These approaches incorporate monolingual data into a machine translation model in different ways. Language model fusion uses monolingual data to train an independent language model. Pre-training and multi-task learning rely on a similar objective function but share the parameters of the language model with the translation model. Similarly, back-translation considers the decoder as a language model where the encoder state is emulated by a synthetic source sentence. Throughout this chapter we analyzed and tuned each of the presented methods individually. Next, we compare the results and consider the question of which method makes the most efficient use of monolingual data. In the following we compare the best-performing methods across all three approaches. We show the result for each language individually in Tables 4.32-4.35, reporting the performance on the development and test set as well the amount of data used for each method.

In particular, all systems reported use 6 layers within both the encoder and 6 the decoder. We use a data ratio of 1:10 for English→Italian and Romanian→English, and for the high-resource tasks of German→English and Chinese→English we use ratios of 1:5 and 1:1 respectively. The pre-training approach that worked best across all tasks relies on a cloze loss on top of the encoder to make use of source data in combination with a language model loss on the decoder which obtains pseudo-source information from a pseudo-encoder. For multi-task training, source information is not helpful and we use solely a language model loss on the decoder to optimize the model on the target monolingual data. In this work, we show that the training of fusion models does not lead to the desired convergence behavior (see Section 4.4.2). The best performance is obtained by log-linear model combination in search with a translation model scale of $\alpha = 1$ and language model scale of $\beta = 0.1$. The model combination is computationally feasible in search with full-size translation models, hence we can use the baseline model directly with six encoder and decoder layers.

Table 4.32 shows the results on the English→Italian task. Back-translation yields the best performance with an improvement of more than +1.5 BLEU [%] and TER [%] on the development and the test set. Using monolingual data to pre-train the encoder and the decoder yields very competitive results; however, the approach uses twelve times more target data as well as an additional monolingual source dataset. Using the language model loss on the decoder with a multi-task learning approach yields weaker results and language model fusion shows the weakest performance, not improving upon the baseline, with a small lead for sequence-level normalized models. Notably, the pre- and multi-task training employ the same monolingual training data as

Table 4.33: Romanian→English: Overview over all methods using monolingual data.

| monolingual data usage | | mono data | | dev | | `newstest2016` | |
|---|---|---|---|---|---|---|---|
| | | src | trg | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| none (baseline) | | - | - | 35.7 | 45.3 | 34.1 | 47.4 |
| back-translation | | - | 0.2G | 37.9 | 43.9 | 36.8 | 45.2 |
| new loss | pre-training (enc, dec) | 0.1G | 1.4G | 37.1 | 44.5 | 35.4 | 46.5 |
| | multi-task (dec) | - | 1.4G | 36.0 | 45.2 | 34.6 | 47.2 |
| lm fusion | sequence normalization | - | 1.4G | 35.8 | 45.8 | 33.8 | 48.5 |
| | symbol normalization | - | 1.4G | 35.4 | 46.0 | 33.8 | 48.0 |

Table 4.34: German→English: Overview over all methods using monolingual data.

| monolingual data usage | | mono data | | dev | | `newstest2017` | |
|---|---|---|---|---|---|---|---|
| | | src | trg | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| none (baseline) | | - | - | 32.7 | 48.6 | 33.6 | 48.0 |
| back-translation | | - | 0.6G | 35.1 | 45.8 | 36.6 | 44.8 |
| new loss | pre-training (enc,dec) | 2.4G | 1.0G | 33.1 | 48.1 | 34.1 | 47.5 |
| | multi-task (dec) | - | 1.0G | 33.1 | 48.4 | 34.1 | 47.9 |
| lm fusion | sequence normalization | - | 1.0G | 33.4 | 49.1 | 34.1 | 48.8 |
| | symbol normalization | - | 1.0G | 33.2 | 49.0 | 34.3 | 48.2 |

the target-side language model.

The experiments on Romanian→English presented in Table 4.33 are generally consistent with these observations. However, back-translation obtains a notable improvement over pre-training, which could be a result of the small amount of monolingual Romanian data.

German→English and Chinese→English are both high-resource tasks providing significantly more bilingual data. In particular, pre-training yields only small improvements, performing about 0.5 BLEU [%] and TER [%] over the baseline for German→English and on par with the baseline for Chinese→English. Pre-training, multi-task learning and language model fusion show very similar results, all ranging between 34.1 and 34.3 BLEU [%] on the test set of the German→English task. Similarly, the results on the Chinese→English translation task are between 23.2 and 23.6 BLEU [%], all improving upon the baseline by a very small margin. Back-translation is the only method to systematically beat the baseline, with +3.0 BLEU [%] and +1.2 BLEU [%] on the test sets of German→English and Chinese→English respectively.

It should be noted that due to technical limitations with respect to resource usage and model training time, the back-translation experiments use the least amount of monolingual data among all methods. Since the target data used for back-translation is a subset of the monolingual data used in all other approaches, the results on back-translation are a lower bound to the performance of a system trained on all possible back-translated data. Furthermore, we observe that adding more synthetic data on the English→Italian and Romanian→English tasks yields diminishing returns. The improvements through back-translation start to flatten on both tasks once the number of synthetic sentences doubles the human translations (see Table 4.30). Hence, we believe that the experiments remain comparable and that the conclusions of the comparison are valid.

We conclude that monolingual data is extremely helpful in the training of machine translation models. On low-resource tasks pre-training is a very effective approach, which can also make use of monolingual source data. Overall, back-translation remains the most effective way to use monolingual data, with respect to both translation performance and data efficiency. It yields the best results across the four tasks while using only 7%–50% of the monolingual data.

Table 4.35: Chinese→English: Overview over all methods using monolingual data.

| monolingual data usage | | mono data | | dev | | newstest2017 | |
|---|---|---|---|---|---|---|---|
| | | src | trg | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| none (baseline) | | - | - | 22.2 | 62.4 | 23.0 | 61.9 |
| back-translation | | - | 0.4G | 23.0 | 61.3 | 24.2 | 60.9 |
| new loss | pre-training (enc,dec) | 54.8M | 1.4G | 22.1 | 62.2 | 23.3 | 61.1 |
| | multi-task (dec) | - | 1.4G | 22.6 | 61.7 | 23.7 | 61.1 |
| lm fusion | sequence normalization | - | 1.4G | 22.6 | 64.0 | 23.2 | 63.1 |
| | symbol normalization | - | 1.4G | 22.3 | 63.5 | 23.6 | 62.1 |

Table 4.36: Domain analysis on global-normalized fusion models. Language models are trained using either only the target side of the bilingual data ($\text{LM}_{\text{biling}}$) or monolingual and bilingual target data ($\text{LM}_{\text{mono}}$). The test set printed in bold is the official test set of the corresponding task.

| task | domain of data | | | LM PPL | |
|---|---|---|---|---|---|
| | bilingual | monolingual | test | $\text{LM}_{\text{biling}}$ | $\text{LM}_{\text{mono}}$ |
| En→It | science | news | **science** | 53.4 | 33.4 |
| | | | news | 94.6 | 22.6 |
| De→En | parliament, other | news | parliament | 24.0 | 25.9 |
| | | | **news** | 84.0 | 39.7 |

### 4.4.6 Analysis: Effect of Domain

Whenever different datasets are combined to train a model the origin of the data and with it the underlying domain can play an important role. In the case of monolingual data, this raises two related questions:

(a) Are the improvements from the use of monolingual data solely a domain effect?

(b) How much impact does the domain of the datasets have?

To answer both questions we first consider the domains of all datasets involved as depicted in Table 4.36. Most shared tasks on machine translation rely on monolingual data obtained from news websites. The bilingual training data as well as the test sets, however, originate from a wider range of data sources. From Table 4.36 we observe that a language model $\text{LM}_{\text{biling}}$ trained on the target side of the bilingual data performs decently on the English→Italian task and very well on German→English if the domain of the test set matches the domain of the training data. In comparison, a language model $\text{LM}_{\text{mono}}$ that is trained additionally on the monolingual news data performs especially well on the news test sets, reducing the perplexity to $\frac{1}{2}$ or $\frac{1}{4}$ of the original value. On the test set that matches the domain of the training data (either science or parliament), the monolingual data helps a lot in the case of the English→Italian, where little bilingual data exists.

Overall, we observe that the domain of the data has a huge impact on how well the target side of a test set can be explained given the training data. We test the baseline and all three methods that make use of monolingual data on the test sets from Table 4.36. One of the test sets matches the domain of the bilingual training data while the other matches the domain of the monolingual target data. The results depicted in Table 4.37 show that the domain of the data has a significant impact. On English→Italian all systems that use monolingual data improve over the baseline on both test sets. The biggest difference is observed when using back-translation, where

Table 4.37: Domain effect of all methods. English→Italian test sets are `tst2010` and `newstest2009` to match the data of the bilingual and monolingual training data respectively. Analogously, `europarldev2006` and `newstest2017` are used in the German→English task.

| language | method | mono data | | domain of test set matches domain of | | | |
|---|---|---|---|---|---|---|---|
| | | | | bilingual data | | monolingual data | |
| | | src | trg | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| En→It | baseline | - | - | 29.6 | 52.0 | 21.6 | 61.5 |
| | back-translation | - | 0.1G | 31.4 | 50.5 | 26.5 | 56.7 |
| | lm fusion | - | 1.2G | 30.2 | 52.1 | 22.7 | 61.0 |
| | pre-training | 2.5G | 1.2G | 31.3 | 50.3 | 24.2 | 58.6 |
| De→En | baseline | - | - | 33.3 | 51.6 | 33.6 | 48.0 |
| | back-translation | - | 0.6G | 33.5 | 51.7 | 36.6 | 44.8 |
| | lm fusion | - | 1.0G | 33.3 | 53.2 | 34.1 | 48.8 |
| | pre-training | 2.4G | 1.0G | 33.2 | 51.9 | 34.1 | 47.5 |

the impact increases from +1.8 BLEU [%] to +4.9 BLEU [%] when the test matches the domain of the monolingual data. All methods show bigger improvements over the baseline if the domain of the monolingual data matches the domain of the test set.

For German→English we observe that no improvement is achieved by adding monolingual data when the domain of the test set matches the domain of the bilingual training data. However, if the test set matches the domain of the training data, all methods yield improvements between +0.5 and +3.0 BLEU [%] over the baseline.

We conclude that the impact of monolingual data varies quite heavily depending on the domain of the datasets involved as well as the amount of bilingual training data. For systems with low amounts of bilingual data the addition of monolingual data improves clearly beyond domain adaptation. However, if enough bilingual data is available, preferably from the test set domain, the system does not benefit from monolingual data. This indicates that for high-resource systems monolingual data provides only domain-adaptation effects.

### 4.4.7 Analysis: Combining Methods

Throughout this work, we show several promising methods that use monolingual data to improve machine translation systems. Overall, back-translation yields the most consistent and strongest performance; however, each method has its specific benefits, for example, pre-training is the only approach that integrates monolingual source data. This raises the question of whether it is beneficial to combine the different approaches. From this we hope to answer:

(a) Can we obtain better overall performance?

(b) Do the various approaches extract the same information from the additional data?

Since back-translation shows the strongest performance across all tasks, we combine it with either pre-training or language model fusion.

We report the results on English→Italian in full detail in Table 4.38. Note that the data used for back-translation is a subset of the data used in pre-training or language model training, hence no additional target-side data is reported when back-translation is applied. We observe that pre-training improves an already strong system with back-translation by 0.4 BLEU [%] and TER [%] on the development set and 0.1 BLEU [%] and 0.4 TER [%] on the test set. Extending the joint approach even further by adding an external language model shows no consistent improvement.

Table 4.38: English→Italian: Combining different methods that make use of monolingual data with back-translation.

| back-translation | method | mono data src | mono data trg | dev Bleu [%] | dev Ter [%] | tst2010 Bleu [%] | tst2010 Ter [%] |
|---|---|---|---|---|---|---|---|
| no | baseline | - | - | 28.1 | 54.0 | 29.6 | 52.0 |
| | lm fusion | - | 1.2G | 28.0 | 54.6 | 30.2 | 52.1 |
| | pre-training | 2.5G | 1.2G | 29.6 | 52.5 | 31.3 | 50.3 |
| yes | baseline | - | 0.1G | 29.9 | 52.5 | 31.4 | 50.5 |
| | lm fusion | - | 1.2G | 29.6 | 53.3 | 31.0 | 51.5 |
| | pre-training | 2.5G | 1.2G | 30.3 | 52.1 | 31.5 | 50.1 |
| | + lm-fusion | 2.5G | 1.2G | 29.8 | 53.1 | 31.6 | 50.8 |

Table 4.39: Effect of combining different methods that make use of monolingual data with back-translation on three translation tasks.

| back-translation | method | additional mono data src | additional mono data trg | Ro→En newstest2016 Bleu [%] | Ro→En newstest2016 Ter [%] | De→En newstest2017 Bleu [%] | De→En newstest2017 Ter [%] | Zh→En newstest2017 Bleu [%] | Zh→En newstest2017 Ter [%] |
|---|---|---|---|---|---|---|---|---|---|
| no | baseline | | | 34.1 | 47.4 | 33.6 | 48.0 | 23.0 | 61.9 |
| | lm fusion | | yes | 33.8 | 48.5 | 34.1 | 48.8 | 23.2 | 63.1 |
| | pre-training | yes | yes | 35.3 | 46.3 | 34.1 | 47.5 | 23.3 | 61.1 |
| yes | baseline | | yes | 36.8 | 45.2 | 36.6 | 44.8 | 24.2 | 60.9 |
| | lm fusion | | yes | 37.8 | 44.4 | 37.9 | 44.8 | 24.0 | 62.6 |
| | pre-training | yes | yes | 37.4 | 44.3 | 37.3 | 44.4 | 24.2 | 60.5 |
| | + lm fusion | yes | yes | 37.7 | 44.5 | 37.9 | 44.9 | 24.0 | 62.4 |

For the three remaining language pairs we report results in Table 4.39. For Romanian→English and German→English, pre-training and language model fusion improve strong back-translation systems by 0.6-1.3 Bleu [%]. This is an important observation since back-translation is considered to be the state-of-the-art approach when using monolingual data. Language model fusion and pre-training show similar Bleu results across all tasks; however, language model fusion tends to generate higher Ter scores.

Going back to the original question, we conclude (a) that even strong systems with back-translation data are improved by the newly introduced methods, with pre-training and language model fusion each performing best on different tasks. Furthermore, we can infer (b) that back-translation extracts different information from the monolingual data than both language model fusion and pre-training. Combining all three methods does not provide meaningful improvements, indicating that there is an overlap in the information obtained from pre-training and language model fusion.

Overall, when applied alongside back-translation, pre-training yields more stable results than language model fusion, hence we recommend using pre-training and back-translation when aiming for best-performance translation systems.

### 4.4.8 Comparison to Other Work

Finally, we compare the presented methods to other works from the literature. Commonly, monolingual data is only added as a last step after performing all other optimization approaches, such as a model ensemble, bigger model size or teacher-student learning. However, these methods can already provide a significant boost over a baseline model, making it hard to obtain an even comparison. Furthermore, there is no guideline by shared tasks such as the IWSLT or WMT on how to select monolingual data from the provided corpora, e.g. by specifying the amount of data or a selection criterion. This is problematic because for many language pairs, there is more monolingual data available than can practically be used by most approaches. This means that it is very hard to find strictly comparable results in the literature. When possible, we report both baseline models as well as fully optimized models from the literature.

Table 4.40: Results of the English→Italian IWSLT task. Systems with multilingual data use more bilingual training data but no additional En→It data. The IWSLT 2017 task allows the use of En ↔ {De,Nl,Ro} data.

| | extra | tst2010 | | tst2017 | |
|---|---|---|---|---|---|
| | data | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| transformer [Lakew & Lotito+ 17a] | multiling | 28.5 | | | |
| Kyoto IWSLT [Dabre & Cromierès+ 17] | multiling | 29.1 | | 30.8 | 50.5 |
| FBK IWSLT [Lakew & Lotito+ 17b] | | | | 29.9 | |
| + Multilingual | multiling | | | 29.6 | 50.7 |
| KIT IWSLT [Pham & Sperber+ 17] | multiling | | | 32.0 | 48.4 |
| baseline (this work) | | 29.6 | 52.0 | 31.9 | 48.1 |
| lm fusion | trg | 30.2 | 52.1 | 32.5 | 47.9 |
| back-translation | trg | 31.4 | 50.5 | 34.4 | 46.1 |
| pre-train | src+trg | 31.3 | 50.3 | 33.9 | 46.6 |
| + back-translation | src+trg | 31.5 | 50.1 | 34.8 | 45.9 |
| + lm fusion | src+trg | 31.6 | 50.8 | 34.4 | 46.4 |

The results on the English→Italian task are shown in Table 4.40. For this task additional multilingual data from the same domain as the training and test set is provided by the IWSLT and commonly used. We show that the use of out-of-domain monolingual data can achieve even better results.

Table 4.41: Results of Romanian→English WMT 2016 task and comparison to previous works.

| | extra | `newstest2016` | |
|---|---|---|---|
| | data | BLEU [%] | TER [%] |
| RNN attention [Sennrich & Haddow+ 16a] | | 29.2 | |
| + back-translation | trg | 33.3 | |
| transformer [Kasai & Cross+ 20] | | 34.5 | |
| transformer [Conneau & Lample 19] | | 28.4 | |
| + back-translation | trg | 34.4 | |
| + pre-training (lm loss) | trg | 31.5 | |
| + pre-training (cloze) | trg | 35.3 | |
| + back-translation + multilingual | trg | 38.5 | |
| this work (baseline) | | 34.1 | 47.4 |
| + back-translation | trg | 36.8 | 45.2 |
| + lm fusion | trg | 33.8 | 48.5 |
| + back-translation | trg | 37.8 | 44.4 |
| + pre-training | src+trg | 35.3 | 46.3 |
| + back-translation | src+trg | 37.4 | 44.3 |
| + lm-fusion | src+trg | 37.7 | 44.5 |

On the Romanian→English task we specifically point out the work on pre-training performed with a language model or cloze loss [Conneau & Lample 19]. This method was developed and published in parallel to and independently of our work. The underlying methodology is very similar, applying a language model or cloze loss to use monolingual data in a separate pre-training step. The resulting parameters provide an initialization for the machine translation model. However, the two works differ in the architecture selected. In this work, we use a transformer with an encoder and decoder, which, as of this writing, is the state of the art for machine translation. Based on this architecture we adapt the pre-training to initialize the parameters of the model. Conneau and Lample use a self-attentive or transformer language model, basically merging encoder, decoder and cross-attention into one component [Conneau & Lample 19]. Our work agrees with the reported results regarding the use of pre-training, reaching an identical performance of 35.3 BLEU [%]. However, we do not confirm their finding that pre-training improves the baseline by 6.9 BLEU [%] and even outperforms back-translation. In both cases we obtain much stronger performance than the rival system, leading us to the conclusion that pre-training has a positive impact, which is, however, smaller than reported. Similar to Conneau and Lample, we observe strong improvements when combining pre-training with back-translation. Please note that their strongest system reported is multilingual, meaning that an exact comparison is infeasible here.

Table 4.42: Results of German→English WMT task and comparison to previous works.

| | extra | dev | | newstest2017 | |
|---|---|---|---|---|---|
| | data | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| sampling-based back-translation [Kim 22] | trg | | | 36.5 | 49.7 |
| + pre-training | trg | | | 36.7 | 49.3 |
| zero-order direct HMM [Bahar 22] | | | | 33.7 | 54.9 |
| zero-order direct HMM [Wang 23] | | 32.3 | 49.5 | 33.8 | 49.5 |
| first-order direct HMM [Wang 23] | | 32.5 | 49.2 | 33.8 | 49.4 |
| transformer [Tang & Müller+ 18] | | | | 33.7 | |
| RWTH 2018 [Schamper & Rosendahl+ 18] | bil+trg | 37.5 | 49.1 | 39.9 | 47.6 |
| Cambridge 2018 [Stahlberg & de Gispert+ 18] | bil+trg | 36.5 | | 38.4 | |
| this work (baseline) | | 32.7 | 48.6 | 33.6 | 48.0 |
| + back-translation | trg | 35.1 | 45.8 | 36.6 | 44.8 |
| + lm-fusion | trg | 33.4 | 49.1 | 34.1 | 48.8 |
| + back-translation | trg | 36.2 | 45.9 | 37.9 | 44.8 |
| + pre-training | src+trg | 33.1 | 48.1 | 34.1 | 47.5 |
| + back-translation | src+trg | 35.5 | 45.5 | 37.3 | 44.4 |
| + lm fusion | src+trg | 35.6 | 46.6 | 37.9 | 44.9 |

The results for German→English are shown in Table 4.42 and for Chinese→English in Table 4.43. We specifically point out the work of Kim which investigates improved methods of generating back-translation data since it also focuses on the use of monolingual data [Kim 22]. Instead of a deterministic search, the back-translation system uses the model distribution to sample during search and provide synthetic source sentences with increased linguistic variety. We observe that the methods presented here yield better results, however, they access greater amounts of monolingual data. Please note that the work by Kim overlaps with this thesis in some chapters and that both authors investigated the impact of pre-training jointly.

Most works from the literature use monolingual data only via back-translation and in combination with many refinements, such as model ensembles, fine-tuning and greatly increased model sizes. Nonetheless, we report competitive results for both tasks, with the exception of models that use additional bilingual data [Schamper & Rosendahl+ 18, Stahlberg & de Gispert+ 18] or bigger models with ensembling [Wang & Gong+ 18, Wang & Li+ 18].

Table 4.43: Chinese→English.

| | extra | `newstest2017` | | `newstest2018` | |
|---|---|---|---|---|---|
| | data | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| RNN [Sennrich & Haddow+ 16a] | trg | 22.9 | | | |
| + ensemble | trg | 25.7 | | | |
| NiuTrans [Wang & Li+ 18] | trg | | | 25.1 | |
| + ensemble, bigger model, re-ranking | trg | | | 28.5 | |
| transformer 'big' [Kasai & Peng+ 21] | | 24.2 | | | |
| transformer [Wang & Gong+ 18] | | 24.3 | | | |
| + back-translation, teacher, ensemble | trg | 27.6 | | | |
| zero-order direct HMM [Wang 23] | | 23.2 | 59.5 | 23.8 | 58.1 |
| first-order direct HMM [Wang 23] | | 23.3 | 59.3 | 23.7 | 58.2 |
| zero-order direct HMM [Bahar 22] | | 23.1 | 61.7 | 23.6 | 65.7 |
| this work (baseline) | | 23.0 | 61.9 | 23.3 | 61.9 |
| + lm-fusion | trg | 23.2 | 63.1 | 23.8 | 63.1 |
| + back-translation | trg | 24.0 | 62.6 | 24.5 | 63.1 |
| + pre-training | src+trg | 23.3 | 61.1 | 23.8 | 61.2 |
| + back-translation | src+trg | 24.2 | 60.5 | 24.7 | 60.7 |
| + lm fusion | src+trg | 24.0 | 62.4 | 24.9 | 62.5 |

### 4.4.9 Conclusion

For many languages, monolingual data can be easily obtained and it commonly dwarfs the amount of bilingual data available. In this chapter, we investigated the questions of whether and how monolingual data can be used to improve a neural machine translation system and concluded that the use of monolingual data provides strong benefits across all tasks.

First, we showed that additional monolingual helps greatly to explain an unseen test set through a language model (Table 4.4). This indicates that even for high-resource tasks, there is not enough bilingual data available to fully learn the target language within the internal language model of the decoder. This is a prerequisite for the viability of any approach that integrates monolingual data.

Second, we proposed two new approaches incorporating monolingual data into a state-of-the-art machine translation system and compared them to the established method of back-translation, i.e. the creation of synthetic data from a human-generated target sentence. All three methods showed improvements on several tasks, proving that monolingual data improves machine translation systems (Tables 4.32-4.35).

The two approaches proposed in this work make use of monolingual data either via language model fusion or through pre-training. Language model fusion is a log-linear model combination between a language and a translation model that requires explicit normalization to produce a probability distribution. In this work, we considered both sequence- and symbol-level normalization and applied the method during search and training. We observed the best performance using a strong language model trained on monolingual target data in the log-linear model combination together with the machine translation model and obtained improvements of around 0.5 BLEU [%] (Table 4.11). Overall, sequence-level normalization showed a slightly stronger performance compared to symbol-level normalized systems. Furthermore, we verified that any improvement gained from the language model integration indeed stems from the use of additional data and is not an effect of model combination or an increased number of parameters (Table 4.14).

Integrating the language model into the training of the machine translation model does not provide additional improvements (Table 4.18) and showed problematic convergence behavior in the desired metrics (Figure 4.8). This could be caused by either a mismatch between the loss of the training and the final metrics of the translation task or an ill-fitting approximation of the loss, namely its denominator.

Pre-training integrates monolingual data without the need for an external model. Instead, parts of the machine translation model are optimized as sub-models to represent the monolingual data and the resulting parameters are transferred back to the machine translation model as an initialization point for the training. We proposed different methods to pre-train the parameters of the encoder, decoder or full translation system using either source or target monolingual data. Using source data to pre-train the encoder we reported improvements of 0.3-1.3 Bleu $^{[\%]}$ (Table 4.21 and 4.22). This is an important finding since, to the best of our knowledge, no established method makes efficient use of monolingual source data, and we showed that the encoder profits both from an explicit loss as well as from additional data (Table 4.25). Using the same loss as in pre-training for multi-task training of a machine translation system did not improve the performance (Table 4.32). Overall, pre-training yields strong results and the best results are obtained if the full system is pre-trained using source and target data to define a loss on the encoder and decoder respectively. We reported improvements of up to 1.3-1.8 Bleu $^{[\%]}$ for the two low-resource tasks (Table 4.26) and only small but consistent improvements of 0.3-0.6 Bleu $^{[\%]}$ on the two high-resource tasks (Table 4.27). This is consistent with similar works, published in parallel, on the same subject [Conneau & Lample 19].

Back-translation is the state-of-the-art approach to include monolingual data. In this work, we used the existing method [Sennrich & Haddow$^+$ 16b] without changes or extensions, by creating a parallel corpus from monolingual target data through the automatic generation of a translation. We verified the results of this approach as we observed an improvement of 1.2-3.0 Bleu $^{[\%]}$ across all tasks (Table 4.30-4.31). Back-translation requires the training of an additional machine translation model as well as the translation of millions of sentences. However, it is overall the most data-efficient method, yielding the best results in all four language pairs, while using only 7-58% of the monolingual data (Table 4.32-4.35).

We showed that all methods are susceptible to domain effects of the underlying data. If enough bilingual training data matches the domain of the test set, no method improves when monolingual data is included. However, if the test domain is not included in the training data, even high-resource tasks observed an improvement of 3.0 Bleu $^{[\%]}$. All investigated methods were affected by the data domain.

Finally, we combined all methods presented in this work and observed improvements of 0.9-1.3 Bleu $^{[\%]}$ over a system with back-translation on two of the four tasks (Table 4.39).

## 4.5 Individual Contributions

In this section, we list the individual contributions of the author in contrast to the work of colleagues in joint research projects related to this dissertation. In particular, in accordance with §5.6 of the doctoral guidelines of the RWTH Aachen University, we note which publications of the author overlap with the presented work.

All experiments in this thesis were performed by the author unless explicitly stated at the experiment or result (e.g. when citing external works). The author was majorly involved in the development of all approaches, their central ideas and their implementation details. However, all approaches were created and investigated in close cooperation with other researchers:

- **language model fusion:** The core idea originates from Wilfried Michel for the topic of automatic speech translation [Michel & Schlüter$^+$ 20], and applying it to the task of machine translation was planned and developed by Wilfried Michel and the author in equal parts. Implementation of the methods was performed by Nils-Philipp Wynand during his bachelor's thesis, which was closely supervised by Wilfried Michel and the author [Wynands 21]. The theory, approaches and methodology were proposed by the supervisors and implementation details were frequently discussed by all three. The methods and their implementation were applied to automatic speech translation and published [Wynands & Michel+ 22].

- **pre-training:** The central idea and methodology were proposed by the author. The implementation was performed by Arne Nix during his master's thesis supervised by the author and Yunsu Kim together with Shahram Khadivi [Nix 19]. Core ideas, methodology and implementation were discussed between all four involved. Parts of this work are also part of the thesis of Yunsu Kim [Kim 22]. A very similar method was developed and published in parallel [Conneau & Lample 19] without any connection to this work or its author.
- **back-translation:** The concept of back-translation was proposed by Sennrich et al. [Sennrich & Haddow+ 16a]. In this work, the approach is implemented to verify existing results and to provide the current state-of-the-art result on the use of monolingual data. All implementations are performed by the author; however, no scientifically new methods were applied throughout this thesis.

The combination of the different approaches was implemented and performed by the author of this thesis.

# 5. Extensions to the Attention Mechanism

Neural machine translation began to achieve promising results when powerful sequence-to-sequence models were able to capture the full input and output sequence. The first neural sequence models employed two independent recurrent layers in an encoder-decoder framework that does not require limited history assumptions on either the source or the target sentence [Sutskever & Vinyals$^+$ 14]. However, the breakthrough that made neural machine translation the undisputed state-of-the-art came with the introduction of the attention mechanism [Bahdanau & Cho$^+$ 15]. The attention mechanism allows the modeling of a differentiable, target-to-source soft alignment that can be trained jointly with the rest of the translation model. The power of the attention mechanism is highlighted by the fact that two of the most distinguishing features of the state-of-the-art transformer architecture are (i) an increased number of encoder-decoder-attention components and (ii) the replacement of all recursive layers by an additional attention mechanism [Vaswani & Shazeer$^+$ 17]. To distinguish these forms of attention we refer to them as cross-attention and self-attention respectively.

Since attention is such an important feature of state-of-the-art machine translation systems, we focus on it throughout this chapter. In the following, we recapitulate the definition of the attention layer, discuss its limitations and suggest several extensions.

## 5.1 Cross-Attention

In this chapter, we focus on cross-attention, i.e. the attention that connects the encoder and the decoder, as introduced by Bahdanau et al. and modified in the transformer architecture [Bahdanau & Cho$^+$ 15, Vaswani & Shazeer$^+$ 17]. Attention is often categorized as the soft lookup of a query on a series of key-value pairs. Formally the input consists of a sequence of query vectors

$$q_1, \dots, q_i, \dots, q_I \in \mathbb{R}^{d_q}$$

together with a sequence of key-value pairs

$$(k_1, v_1), \dots, (k_J, v_J) \in \mathbb{R}^{d_k} \times \mathbb{R}^{d_v}.$$

The intuition is that for each query $q_i$ we want to find the most similar key $k_j$ and output its corresponding value vector $v_j$. We perform a soft lookup by calculating a similarity measure $\hat{\alpha}$ between the query $q_i$ and all key vectors $k_1^J$. For the output of the soft lookup, each value vector $v_j$ should be considered proportional to the similarity $\hat{\alpha}(q_i, k_j)$ of its key vector $k_j$ to the current query. In the case of machine translation, cross-attention is a sub-layer of the decoder depicted in Figure 5.1, and we use the intermediate states $s_0^I$ of the decoder as queries and the outputs of the encoder $h_1^J$ as key-value pairs, i.e.

$$q_i := s_i \qquad \text{and} \qquad k_j := v_j := h_j.$$

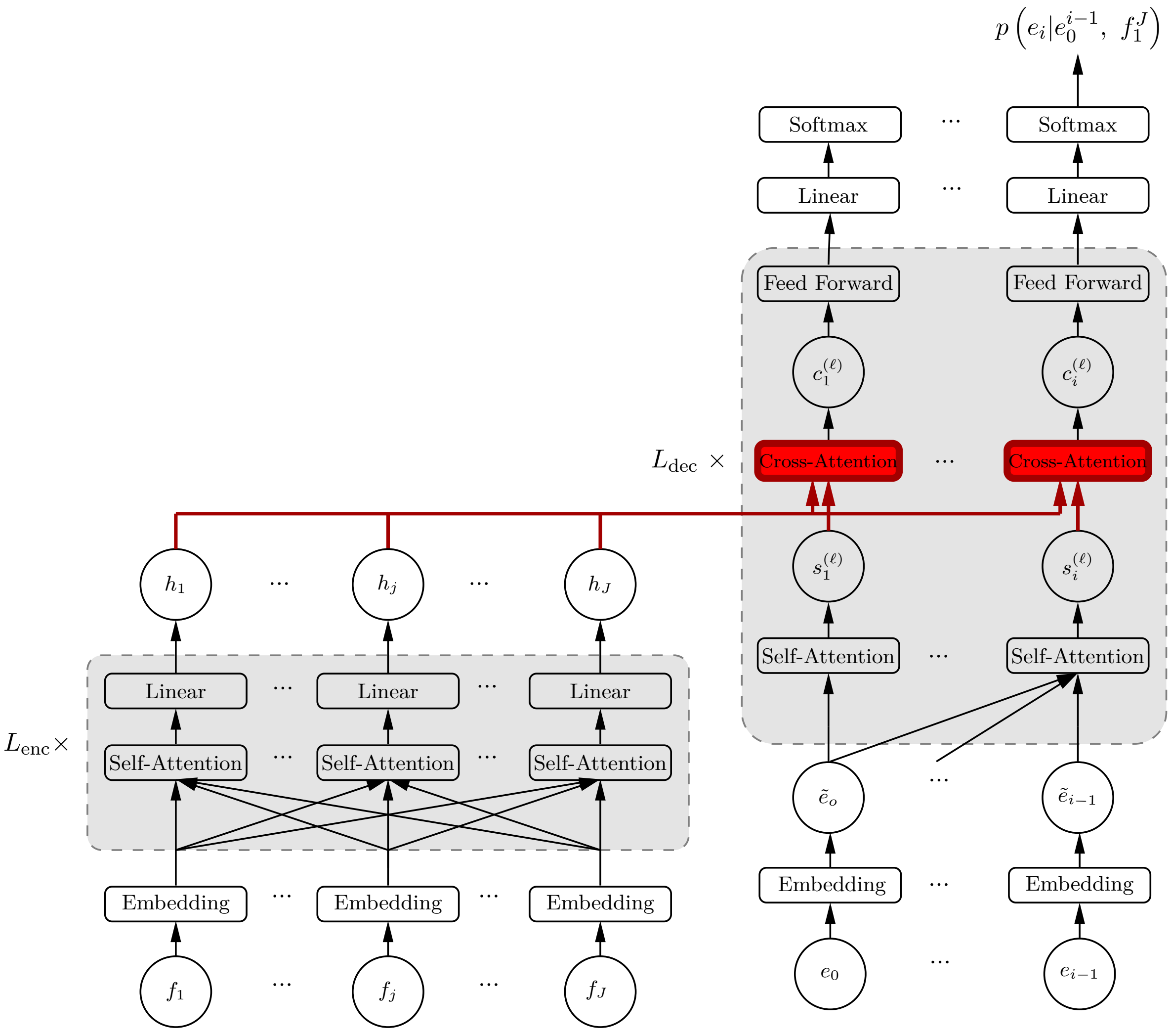


Figure 5.1: Transformer architecture with the cross-attention layer highlighted in red.

In the following, we introduce the cross-attention layer using the machine-translation-specific notations and conventions.

To perform the soft lookup, a similarity $\hat{\alpha}$ between the current query $s_i$ and all keys is calculated

$$\hat{\alpha}_{i,j} := \hat{\alpha}(h_j, s_i) := \frac{1}{\sqrt{d_{\text{att}}}} \left(W_k h_j\right)^\top W_q s_i \quad \in \mathbb{R} \qquad \forall 1 \le j \le J \tag{5.1}$$

which we refer to as *attention energy*. The matrices $W_k, W_q$ are trainable parameters of the network and they project key and query vectors to the same dimension $d_{\text{att}}$. The attention energies are normalized over all the key-value pairs, namely the number of encoder positions $J$, by a softmax operation

$$\alpha(j|i) := \frac{\exp\left(\hat{\alpha}_{i,j}\right)}{\sum_{j'} \exp\left(\hat{\alpha}_{i,j'}\right)} \tag{5.2}$$

and the resulting probability distribution $\alpha(j|i)$, commonly called the *attention weights*, is used to compute the *context vector* at position $i$

$$c_i := \sum_j \alpha(j|i) W_v h_j \tag{5.3}$$

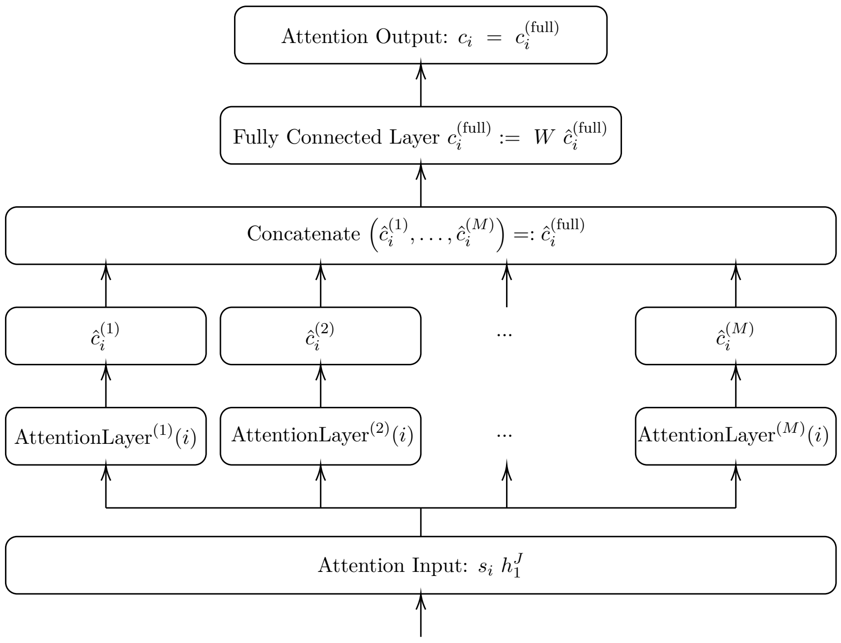

Attention Output: $c_i = c_i^{(\text{full})}$
Fully Connected Layer $c_i^{(\text{full})} := W\ \hat{c}_i^{(\text{full})}$
Concatenate $\left(\hat{c}_i^{(1)}, \ldots, \hat{c}_i^{(M)}\right) =: \hat{c}_i^{(\text{full})}$
$\hat{c}_i^{(1)}$
$\hat{c}_i^{(2)}$
...
$\hat{c}_i^{(M)}$
AttentionLayer$^{(1)}(i)$
AttentionLayer$^{(2)}(i)$
...
AttentionLayer$^{(M)}(i)$
Attention Input: $s_i\ h_1^J$

Figure 5.2: Multi-head cross-attention with query $s_i$ and key-value sequence $h_1^J$.

as a weighted average of the values. Since the similarity $\hat{\alpha}$ of query and key is computed via a product, this variant of attention is often called *multiplicative attention.* This allows for computation- and memory-efficient calculation of the attention via matrix multiplication. To simplify notation, the vectors in the query, key and value sequences are individually concatenated, yielding the matrices

$$\begin{aligned} Q &:= (W_q s_1, \ldots, W_q s_I) \\ K &:= (W_k h_1, \ldots, W_k h_J) \\ V &:= (W_v h_1, \ldots, W_v h_J) \end{aligned}$$

allowing us to express the calculation of the attention weights as

$$(c_1, \ldots, c_I) = V^\mathsf{T} \cdot \text{softmax}\left(\frac{K^\mathsf{T} Q}{\sqrt{d_\text{att}}}\right).$$

Since matrix operations are highly optimized on modern GPUs, the sequence of context vectors can be computed extremely efficiently [Vaswani & Shazeer$^+$ 17].

In the transformer architecture numerous cross-attention operations are performed. In particular, every decoder layer $\ell \in \{1, \ldots, L_\text{dec}\}$ performs a predefined amount of $M$ independent attention operations. Each of these $m \in \{1, \ldots, M\}$ operations is called an *attention head* with its own weight matrices $W_q^{(\ell,m)}, W_k^{(\ell,m)}, W_v^{(\ell,m)}$, allowing them to focus on different parts of the source sentence. While each attention head operates on the encoder outputs $h_1^J$ as key-value pairs, the query vector $s_i^{(\ell)}$ is layer-specific, hence

$$\begin{aligned} \hat{\alpha}_{i,j}^{(\ell,m)} &:= \hat{\alpha}^{(\ell,m)}(h_j, s_i^{(\ell)}) \\ \alpha^{(\ell,m)}(j|i) &:= \frac{\exp\left(\hat{\alpha}_{i,j}^{(\ell,m)}\right)}{\sum_j \exp\left(\hat{\alpha}_{i,j}^{(\ell,m)}\right)} \\ c_i^{(\ell,m)} &:= \sum_j \alpha^{(\ell,m)}(j|i)\ W_v^{(\ell,m)} h_j. \end{aligned}$$

Multi-head attention is visualized in Figure 5.2. For simplicity of notation, we drop the dependency on the head and decoder layer for the remainder of this work and each modification is applied to all cross-attention heads and layers.

## 5.2 Extending the Dependencies

In order to improve the attention mechanism, we extend the information accessible within the layer. The baseline attention weights for the position pair $(j, i)$ are calculated from the sequence of all encoder states $h_1^J$ and the current decoder state $s_i$ at position $i$

$$\alpha(j|i) = \alpha(j|h_1^J, s_i).$$

These attention weights are obtained via normalization from the attention weights

$$\hat{\alpha}_{i,j} = \hat{\alpha}(h_j, s_i).$$

Note that the energies only depend on a single encoder output $h_j$ of the source position $j$. We argue that the dependencies of $\hat{\alpha}$ provide a more meaningful description of the overall dependencies of the attention layer since the full encoder output $h_1^J$ is used in the attention weight $\alpha(j|i)$ in a rather limited fashion. For example, the order of the encoder outputs is completely irrelevant to the attention weight calculation $\alpha(j|i)$ at position $j$. The only information used from the other encoder positions $j' \neq j$ is their exponential sum, meaning that even the number of encoder outputs $J$ is only incorporated implicitly. Any modification to the attention energies that does not alter this sum will result in the same attention weight $\alpha(j|i)$ for the encoder position $j$. Hence, if we discuss the dependencies of the attention layer, we will consider the dependencies of its energies $\hat{\alpha}$.

In the following, we will consider different extensions to the attention mechanism, each adding further dependencies.

### 5.2.1 Word Embeddings within Encoder and Decoder

The attention mechanism is frequently seen as a replacement of the target-to-source alignment model in count-based machine translation systems [Bahdanau & Cho+ 15, Alkhouli & Bretschner+ 18, Garg & Peitz+ 19]. An alignment is a mapping that assigns a source word to each target word, following the intuition that the existence of each target word can be explained by a corresponding source word. However, unlike traditional alignment models, the words under consideration are not a direct input to the attention layer. Instead of the embedding $\tilde{f}_j$ of the word $f_j$, the attention layer utilizes its contextual encoder representation $h_j$. Intuitively, $h_j$ represents the word $f_j$; however, the actual flow of information from the embedded words $f_1^J$ to a certain encoder state $h_j$ is unknown. Unlike recurrent architectures, the path from all input words $f_1, \ldots, f_J$ through the transformer encoder to a specific $h_j$ is the same length. This can be seen as a strength of self-attention layers as it facilitates long-dependency learning [Vaswani & Shazeer+ 17]; however, the connection between word $f_j$ and encoder state $h_j$ is obscured. Similarly, the current decoder state $s_i$ has no immediate association with the last target word $e_{i-1}$.

A straightforward solution providing the attention layer with information about the corresponding source or target word is the concatenation of the corresponding word embedding $\tilde{f}_j$, respectively $\tilde{e}_{i-1}$. This yields new key and query vectors

$$h_j^{(\text{new})} := \begin{pmatrix} h_j \\ \tilde{f}_j \end{pmatrix}, \qquad s_i^{(\text{new})} := \begin{pmatrix} s_i \\ \tilde{e}_{i-1} \end{pmatrix}$$

which supply new inputs to the cross-attention layer with strong ties to a certain word. We provide either the original embeddings $\tilde{f}_1^J$ or a layer-normalized version $\mathrm{LayerNorm}(\tilde{f}_1^J)$. Overall, the approach is similar to a residual or skip connection from the word embedding $f_j$ around the encoder.

Instead of applying a concatenation, the new information can also be included via addition

$$h_j^{(\text{add})} := h_j + \tilde{f}_j \qquad\qquad s_i^{(\text{add})} := s_i + \tilde{e}_{i-1}$$

where $s_i^{(\text{add})}$ is used as query vector to the sequence of key and value vectors $h_1^{(\text{add})}, \ldots, h_J^{(\text{add})}$. To calculate the output of the attention layer, the first operation applied to the input vectors is a linear transformation (see Equations 5.1 and 5.3). In the case of the query vector $s_i^{(\text{add})}$ the multiplication with the weight matrix $W_q$ yields

$$\begin{aligned} W_q s_i^{(\text{add})} &= W_q s_i + W_q \tilde{e}_{i-1} \\ &= \begin{pmatrix} W_q & W_q \end{pmatrix} \cdot \begin{pmatrix} s_i \\ \tilde{e}_{i-1} \end{pmatrix} \\ &= W_q^{(\text{new})} s_i^{(\text{new})}. \end{aligned}$$

We conclude that providing additional information to the query by addition is a special case of concatenating the same information. This observation holds in general when combining hidden states of a model, provided that the next operation is a linear transformation, as is the case for attention and linear layers. Since these are the core layers of the transformer, we only consider the concatenation of new information for all upcoming experiments.

In total, we obtain the following new dependencies for this modification of the attention mechanism

$$\hat{\alpha}_{i,j} = \hat{\alpha}(h_j, \tilde{f}_j, s_i, \tilde{e}_{i-1}).$$

Note that, strictly speaking, we are not adding new information to the attention mechanism, since the source and target word embeddings are implicitly included in the encoder outputs $h_1^J$, respectively the decoder state $s_i$. However, this information is not explicitly present in the baseline attention. Next, we consider extensions that provide the attention layer with information that cannot be derived from the existing inputs.

### 5.2.2 Higher-Order Assumption in Attention Layers

Attention layers proved to be an extremely effective component of machine translation models. Functionally they replace the explicit alignment model used in traditional count-based systems [Bahdanau & Cho$^+$ 15]. Notably, attention layers are a feed-forward structure [Vaswani & Shazeer$^+$ 17] and replacing the recurrent architecture [Bahdanau & Cho$^+$ 15] with self-attention allows for the parallel computation of all encoder steps. This speed-up compared to recurrent architectures is a significant part of the success of the transformer architecture [Chen & Firat$^+$ 18]. However, the calculation of the cross-attention energies

$$\hat{\alpha}_{i,j} = \hat{\alpha}(h_j, s_i)$$

relies solely on the encoder output and the current decoder state. Crucially, this means that no information about previous attention decisions of the model is available in the attention layer. This stands in contrast to various count-based alignment models which use first-order dependencies [Vogel & Ney$^+$ 96, Och & Ney 03]. The information about previous attention decisions could be helpful to model locality, i.e. the assumption that the next source word to translate tends to be positionally close to the last translated word. Of course, such a property is highly dependent on the language pair under consideration. But even among completely different languages, for which translation can require a lot of reordering, the translation of sub-words is expected to be locally monotonous. For example, when translating a source word, which is split into separate sub-words by BPE, we assume that the system keeps focusing on consecutive sub-words until the current word is fully translated.

Another important issue concerns the modeling of coverage and over-translation. Intuitively, each part of the source sentence should be translated, and once this is done it should not be translated again. This requirement can be captured by alignment models that require a source word to be aligned to a target word (*coverage*) but not arbitrarily many (*over-translation*). The exact boundaries of word alignment are unclear since not every source word is translated literally to exactly one target word. A figure of speech in the source language might be translated to a phrase in the target sentence that does not allow for a meaningful word-to-word alignment. With that in mind, the general intuition remains strong that each word in the target sentence should be explained by a word or a sequence of words in the source sentence. This explanation can provide a meaningful tool to ensure coverage and prevent over-translation. However, to model coverage, the translation system needs to keep track of what parts of the source sentence have already been translated. It is unclear whether that is possible in the current cross-attention layer since it only obtains information about the previous target words $e_1^{i-1}$ and has no access to previous attention decisions $\alpha$ at all. Thus the attention energies $\hat{\alpha}_{i,j}$ at decoder step $i$ cannot focus on encoder positions that were ignored in previous decoding steps, making any modeling of coverage problematic.

Throughout this chapter, we suggest various changes to create a higher-order cross-attention layer by including information about previous attention steps. Focusing mainly on first-order dependencies, we consider variations with a higher-order dependency.

We are faced with the questions of how to encode the desired information and where to place it in the attention layer. First, we consider a series of changes based on an extended query vector

$$s_{i,j}^{(\text{new})} := \begin{pmatrix} s_i \\ \gamma_{i,j} \end{pmatrix} \tag{5.4}$$

and investigate different options for the value of $\gamma_{i,j}$. Note that we allow for a query vector $s_{i,j}^{(\text{new})}$ that varies across different source positions $j$. We use $\gamma_i$ if $\gamma_{i,j} = \gamma_{i,j'}$ for all $1 \leq j, j' \leq J$. Like all decoder states, $\gamma_{i,j} = \gamma_{i,j}^{(\ell,m)}$ is head- and layer-specific and we simplify the notation wherever possible. In the following, we present different approaches to modeling $\gamma_{i,j}$.

### Extending the Query with the Previous Context Vector

The context vector $c_i$ is the output of the attention layer at position $i$, and in principle it compresses all information about hidden states within the layer. Hence, the previous context vector $c_{i-1}$ can be considered an adequate representation of the previous attention step and a suitable choice for $\gamma_{i,j}$. We experiment with using either the previous, head-specific context vector

$$\gamma_i^{(\ell,m)} = c_{i-1}^{(\ell,m)}$$

or the previous full context vector $c_{i-1}^{(\ell)}$, which is obtained from the concatenation of all head-specific context vectors

$$\begin{aligned}\gamma_i^{(\ell)} =& W_{\text{att}} \cdot c_{i-1}^{(\ell)} \\ :=& W_{\text{att}} \cdot \begin{pmatrix} c_{i-1}^{(\ell,1)} \\ \vdots \\ c_{i-1}^{(\ell,M)} \end{pmatrix}\end{aligned}$$

with linear transformation $W_{\text{att}}$. For the first decoder step no previous context vector is available and we initialize $\gamma_0^{(\ell)} := 0$ for all layers.

In total, we obtain attention energies

$$\hat{\alpha}_{i,j} = \hat{\alpha}(h_j, s_i, c_{i-1}) \tag{5.5}$$

with first-order dependency.

**Extending the Query with the Previous Attention Weight**

The context vector $c_{i-1}$ aggregates all the information of the attention layer. It is directly calculated from and contains information about the attention weights and thus the alignment from the previous decoder step. However, this information is rather indirect, since the attention weights are only present as the scalar weights in the calculation of the weighted sum

$$c_{i-1} = \sum_{j=1}^{J} \alpha(j|i-1) W_v h_j.$$

Providing all attention weights from the previous decoder step $\gamma_{i,j} := (\alpha(1 \mid i-1), \ldots, \alpha(J \mid i-1))$ directly is problematic in practice since their number depends on the source length $J$ and is not a constant value. The architecture of the attention mechanism does not allow keys to be of variable length since they are multiplied by a fixed-size matrix $W_k$. Instead of providing all attention weights of the previous decoder step at once, we consider each encoder output $h_j$ independently and only concatenate the corresponding attention weight $\alpha(j|i-1)$

$$\gamma_{i,j} := \alpha(j|i-1).$$

This leads to a key-specific query $s_{i,j}^{(\text{new})}$, i.e. a query that is different for each encoder position $j$.

The dependencies of the resulting first-order attention model

$$\hat{\alpha}_{i,j} = \hat{\alpha}(h_j, s_i, \alpha(j|i-1)) \tag{5.6}$$

for the attention energies and

$$\alpha(j \ | h_1^J, s_i, \alpha(1|i-1), \ldots, \alpha(J|i-1))$$

for the attention weights. Since the context vector is directly calculated from the attention weights and the decoder outputs

$$c_{i-1} = c\left(h_1^J, \alpha(1|i-1), \ldots, \alpha(J|i-1)\right)$$

the formal dependencies of the attention energies derived from Equations 5.5 and 5.6 are identical.

### Extending the Query with Accumulated Attention Weights

So far we have provided additional information to the attention mechanism from the previous decoder step $i-1$. While such a first-order assumption can be helpful, modeling concepts such as coverage remains challenging. In order to keep track of translated words, we supply the attention layer with the accumulated attention weights from all previous time steps

$$\gamma_{i,j} = \sum_{\hat{i}=1}^{i-1} \alpha(j|\hat{i}). \tag{5.7}$$

Intuitively, this measures to what degree the current source position $j$ is already translated in the previous $i-1$ decoder steps. The attention energies are normalized over the number of source positions $J$ and generally not over the target $I$, thus source positions can obtain an accumulated attention value $\gamma_{i,j}$ bigger than one.

### Extending the Query with Fertility

The sum of the accumulated attention weights (Equation 5.7) indicates for each source position $j$ to what degree the encoder output $h_j$ is used to generate the partial target sentence $e_1^{i-1}$. However, this assumes that each source position contributes roughly equally to the translation. In contrast to this Brown et al. assigned each source word $f_j$ a fertility $\varphi_j \in \mathbb{N}$, i.e. a scalar value that describes to how many words it translates [Brown & Cocke$^+$ 90]. This idea is extended to neural networks [Tu & Lu$^+$ 16], where the fertility $\varphi_j$ is predicted via a fully connected layer applied to the encoder output $h_j$

$$\varphi_j := \varphi_{\max} \cdot \sigma\left(v^\mathsf{T} h_j\right)$$

with hyperparameter $\varphi_{\max} \in \mathbb{N}$ and trainable parameters $v$. The sigmoid function $\sigma$ maps the output of the fully connected layer to the range of $[0,1]$, and scalar multiplication with $\varphi_{\max}$ projects this to the interval $[0, \varphi_{\max}]$. The fertility is used to scale down the accumulated attention weights

$$\gamma_{i,j} = \frac{1}{\varphi_j} \sum_{\hat{i}=1}^{i-1} \alpha(j|\hat{i}).$$

This means that $\gamma_{i,j} < 1$ implies an under-translated source position $j$ and $\gamma_{i,j} > 1$ indicates that the source position $j$ has already received more attention energy than predicted by the fertility and should not be considered for $i' > i$. Because the fertility $\varphi_j = \varphi(h_j)$ is a property of a source word $f_j$, we calculate it once per encoder output and share it across all layers and attention heads of the decoder $\varphi_j^{(\ell,m)} = \varphi_j$.

We want to point out some important differences between the original fertility model [Brown & Cocke$^+$ 90] and the proposed variant suitable for neural machine translation. Originally fertility was a natural number assigned to each source word $f_j$. In contrast to this, all state-of-the-art neural machine translations operate on a sub-word level and consequently the encoder outputs $h_j$ and fertility values $\varphi_j$ correspond to source sub-words. It remains an open question to what degree the concept of fertility is applicable and whether each chunk of a split-up word needs to be explicitly translated to certain parts of the target sentence. Notably, the fertility values in a neural network do not need to be natural numbers, allowing for real values instead. This could be used to overcome the problems of a word with an original fertility of one being split into three sub-words. Since none of the sub-words describes the original word completely, a naturally valued fertility model would either need to drop information or risk over-translating. Fractional fertilities

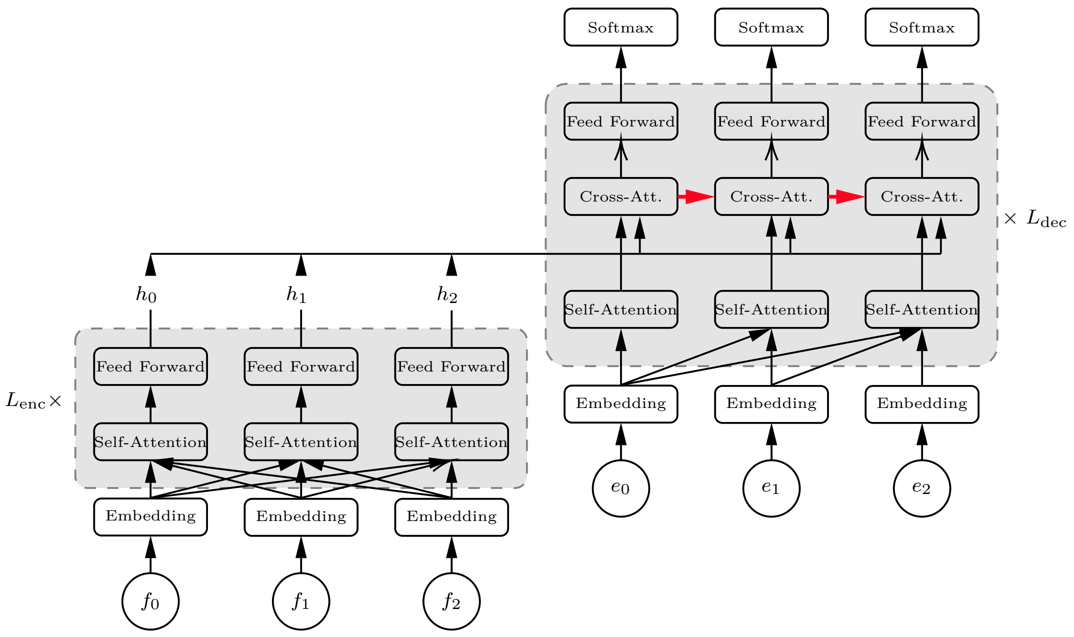


Figure 5.3: Transformer with first-order cross-attention. New dependencies (highlighted in red) introduce recurrency to the decoder.

allow weighing each sub-word according to its importance to the original word and in consistency with the original fertility. Since we compute the fertility from the encoder outputs $h_j$ instead of the word embeddings $\tilde{f}_j$, the current context of a sub-word is taken into account.

In total, we obtain attention energies

$$\hat{\alpha}_{i,j} = \hat{\alpha}\left(h_j, s_i, \sum_{\hat{i}=0}^{i-1} \alpha(j|\hat{i})\right)$$

with the dependencies extended to include all previous attention decisions.

## Theory

Integrating information from the previous attention steps results in a first-order attention model. From a modeling perspective this is desirable as additional inputs can lead to more expressive models. However, from a technical perspective the extended dependencies provide a challenge since they turn the attention mechanism into a recurrent layer, as depicted in Figure 5.3. Recurrent architectures are well-investigated for the task of machine translation [Sutskever & Vinyals$^+$ 14, Bahdanau & Cho$^+$ 15] and achieved good results in the past. The transformer, on the other hand, is a purely feed-forward architecture allowing for more parallelized training. For example, in Figure 5.3 all self-attention sub-layers within the first decoder layer can be computed in parallel once the embedding layer is completed. In contrast to this, the cross-attention layer of decoder layer $\ell$ at decoder time step $i$ cannot be calculated before the previous cross-attention at time step $i-1$ is computed. This recurrence slows down the training of the translation architecture and we investigate the impact on experimental results in Section 5.4.2. The slowdown of the training is problematic because the amount of training time is a restricting factor for machine translation systems. Furthermore, some research suggests that the main benefit of the transformer is its efficiency with respect to training time [Chen & Firat$^+$ 18, Zeyer & Bahar$^+$ 19].

In this chapter, we have proposed three extensions to the attention mechanism that incorporate additional information by expanding the query vector with a representation $\gamma_{i,j}$

$$s_i^{(\text{new})} = \begin{pmatrix} s_i \\ \gamma_{i,j} \end{pmatrix}.$$

In the following, we investigate how this modification impacts the full attention calculation. Starting with the attention energies as defined in Equation 5.1, we assume a $\gamma_i = \gamma_{i,j}$ independent of the encoder position $j$ and obtain

$$\begin{aligned}\hat{\alpha}_{i,j}^{(\text{new})} &= \frac{1}{\sqrt{d_{\text{att}}}} \left(W_k h_j\right)^\intercal W_q \begin{pmatrix} s_i \\ \gamma_i \end{pmatrix} \\ &= \underbrace{\frac{1}{\sqrt{d_{\text{att}}}} \left(W_k h_j\right)^\intercal W_q^{(1)} s_i}_{\hat{\alpha}_{i,j}} + \underbrace{\frac{1}{\sqrt{d_{\text{att}}}} \left(W_k h_j\right)^\intercal W_q^{(2)} \gamma_i}_{\hat{\alpha}_{i,j}^{(\gamma)}}\end{aligned}$$

where the weight matrix is split in two sub-matrices $W_q = \left(W_q^{(1)}, W_q^{(2)}\right)$. This means that the resulting attention energy $\hat{\alpha}_{i,j}^{(new)}$ can be expressed as the sum of two attention energies stemming from the two different queries $s_i$ and $\gamma_i$. Notably, the first of these yields the baseline attention energies $\hat{\alpha}_{i,j}$. Considering the final attention weights as calculated by Equation 5.2

$$\begin{aligned}\alpha^{(\text{new})}(j|i) &= \frac{\exp\left(\hat{\alpha}_{i,j}^{(\text{new})}\right)}{\sum_j \exp\left(\hat{\alpha}_{i,j}^{(\text{new})}\right)} \\ &= \frac{\exp\left(\hat{\alpha}_{i,j}\right) \cdot \exp\left(\hat{\alpha}_{i,j}^{(\gamma)}\right)}{\sum_j \exp\left(\hat{\alpha}_{i,j}\right) \cdot \exp\left(\hat{\alpha}_{i,j}^{(\gamma)}\right)}\end{aligned}$$

we observe that extending the query vector by concatenation essentially provides a second attention mechanism that is normalized jointly with the existing version. Thus every attention decision is a compromise of two independent systems, one modeling the current target-to-source alignment and one the alignment history. Next, we propose a method that explicitly models the balance between the baseline attention and the new information.

## Extended Key-Value List

The attention layer receives a series of keys and values as input. Each key $k_j = h_j$ is compared to the active query $s_i$ and the corresponding value vector $h_j$ influences the output of the attention $c_i$ proportionally to its similarity $\hat{\alpha}_{i,j}$. So far we added additional information to the query vector, effectively allowing a re-scaling of the importance of all source positions depending on past attention decisions. An alternative way to provide information about the past attention steps is via a separate key-value pair. This yields new hidden states of the attention layer, and to avoid confusion with the hidden states of the baseline model, we denote e.g. the new context vector $c_i^{(\text{new})}$. We extend the existing key and value sequence by the last context vector $c_{i-1}^{(\text{new})}$

$$K_i := V_i := \left(h_1, h_2, \ldots, h_J, W_{\text{ctx}} c_{i-1}^{(\text{new})}\right)$$

which is transformed via matrix multiplication with $W_{\text{ctx}}$ to match the dimensions of the other keys and values. Since the context vector depends on the decoder time step $i$, the modified sequences of key and value vectors are also dependent on the decoder time step $i$.

This modification allows for a new position to attend to. Thus the attention energies as defined in Equation 5.1 are unchanged for the existing $J$ positions

$$\begin{aligned} \hat{\alpha}_{i,j}^{(\text{new})} &= \hat{\alpha}_{i,j} && j \leq J \\ \hat{\alpha}_{i,J+1}^{(\text{new})} &= \frac{1}{\sqrt{d_{\text{att}}}} \left( W_k W_{\text{ctx}} c_{i-1}^{(\text{new})} \right)^{\mathsf{T}} W_q s_i \end{aligned} \tag{5.8}$$

and after normalization of $\hat{\alpha}^{(\text{new})}$ over all $J+1$ positions, we obtain the new context vector

$$\begin{aligned} c_i^{(\text{new})} &= \left( \sum_{j=1}^{J} \alpha^{(\text{new})}(j|i) W_v h_j \right) + \alpha^{(\text{new})}(J+1|i) W_v W_{\text{ctx}} c_{i-1}^{(\text{new})} \\ &= \pi c_i + \alpha^{(\text{new})}(J+1|i) W_v W_{\text{ctx}} c_{i-1}^{(\text{new})} \end{aligned}$$

with $\pi \leq 1$. This means that the new context vector $c_i^{(\text{new})}$ is a scaled-down version of the baseline context version $c_i$ combined with a linear projection of the previous context vector $c_{i-1}^{(\text{new})}$.

From Equation 5.8 we observe that the dependencies of the attention energies $\hat{\alpha}$ depend on the position $j$

$$\begin{aligned} \hat{\alpha}_{i,j} &= \hat{\alpha}(h_j, s_i) && j \leq J \\ \hat{\alpha}_{i,J+1} &= \hat{\alpha}(c_{i-1}, s_i). \end{aligned}$$

with overall the dependencies for the attention weights

$$\alpha(j|h_1^J, s_i, c_{i-1}).$$

### 5.2.3 Training and Search

All presented variants of the attention mechanism rely on the training criterion of the baseline system presented in Equation 3.6

$$F = \sum_{(f_1^J, e_1^I) \in \mathcal{T}} \log p(e_1^I | f_1^J)$$

as well as the length-normalized decision rule shown in Equation 3.11a

$$\hat{e}_1^{\hat{I}} = \arg\max_{I, e_1^I} \left\{ \sqrt[I]{p(e_1^I | f_1^J)} \right\}$$

during the search.

## 5.3 Related Work

Cross-attention was introduced into neural machine translation systems as a parametrized alignment model that connects the encoder and the decoder [Bahdanau & Cho+ 15]. It quickly became a core component of state-of-the-art machine translation systems, and by 2016 the majority of the best-performing systems in the WMT task on news translation applied attention layers [Bojar & Chatterjee+ 16]. Self-attention and multi-head attention provided an important change in the way attention layers are used [Vaswani & Shazeer+ 17]. Self-attention layers operate on a single sequence; in other words, the query, key and value sequence are identical. They replaced recurrent layers as the primary sequence layers within the encoder and the decoder. Multi-head attention performs attention several times in parallel on the same input sequences but with individual weight matrices. The resulting context vectors are concatenated and passed through a linear layer to form the final output of the layer. For an in-depth description, we refer the reader to Section 5.1. The combination of cross-, self- and multi-head attention in the transformer architecture proved to be extremely effective and the architecture was applied to problems such as speech recognition [Zeyer & Bahar$^+$ 19], image processing [Parmar & Vaswani$^+$ 18] and general (pre-)training for many natural language processing tasks [Devlin & Chang$^+$ 19].

Since the attention layer plays such a critical role in state-of-the-art systems, many modifications and extensions have been investigated. Many works focus on the improvement of attention via additional information. This information can be obtained from external models like an alignment model [Alkhouli & Ney 17] or by providing additional hidden states of the model to the attention layer. In this work, we investigate the second variant and we focus here on the description of such approaches. Feng et al. experiment with an extension similar to this work in the context of cross-attention in recurrent translation architectures. Adding the previous context vector or the accumulated attention weights leads to more than 2.0 Bleu $^{[\%]}$ improvement in the translation performance as well as better alignments [Feng & Liu$^+$ 16]. Fertility and coverage vectors in a similar architecture provide mixed results, with an average improvement of 2 Bleu on two language pairs and no change on two other language pairs [Mi & Sankaran$^+$ 16]. Instead of modeling fertility to measure how often a word is translated, Cohn et al. consider the inverse, measuring how much of a word is left to translate [Cohn & Hoang$^+$ 16]. In the same work, the authors also provide an explicit external alignment obtaining significant improvements on one out of four language pairs. In contrast to this, other investigations show no significant or consistent improvement when using higher-order dependencies in cross-attention layers [Peter 20]. In this work, we continue the ongoing debate on whether higher-order cross-attention layers are beneficial. Notably, all these investigations were performed using a recurrent translation system with a single cross-attention layer. This is in contrast to state-of-the-art models, which employ a multi-head cross-attention layer in each decoder layer, typically resulting in at least 48 cross-attention layers. Since multiple investigations have shown that different attention heads within and across layers perform different tasks [Voita & Talbot$^+$ 19, Michel & Levy$^+$ 19, Zhang & Yu$^+$ 20], these discrepancies in architectures might have a significant impact on how the attention modifications affect the overall translation system. Furthermore, all described works employ an additive attention mechanism, while state-of-the-art systems rely on multiplicative attention, which can show different behavior [Britz & Goldie$^+$ 17].

In the transformer architecture, i.e. an architecture with multi-head, multi-layer cross-attention, many extensions to the attention layers have been suggested. Adding additional information in the form of an external alignment does not benefit the global translation performance but improves the translation of words from a pre-defined dictionary [Alkhouli & Bretschner$^+$ 18]. Most works focus on the self-attention layer, e.g. by providing explicit positional information to the key and value of the self-attention layer. This improves the performance on language modeling tasks [Dai & Yang$^+$ 19] as well as on the English→German translation task [Shaw & Uszkoreit$^+$ 18], mostly due to better length generalization [Rosendahl & Tran$^+$ 19]. In these approaches the distance between the query position and the key position is forwarded to the self-attention layer. It is not clear how to generalize these approaches to cross-attention since this relative positional information relies on an implicit sequence alignment which does not generalize if the key and query sequences vary in length and order. Replacing the probability distribution of the self-attention weights by a fixed Gaussian distribution around the current position does not change the translation performance on seven out of eight language pairs [You & Sun$^+$ 20]. This means that reducing the dependency of attention weights down to just the current query position does not harm the self-attention layer. However, applying hard-coded Gaussian attention to the cross-attention yields a decrease of 5-10 Bleu $^{[\%]}$ across all language pairs. We conclude that cross-attention is more reliant on

Table 5.1: Adding word embeddings to the encoder output $h_j$ or the decoder state $s_i$. All results reported on the development set.

| | En→It | | Ro→En | |
|---|---|---|---|---|
| | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| baseline | 28.1 | 54.0 | 35.7 | 45.3 |
| source embedding | 27.0 | 55.2 | 35.7 | 45.5 |
| + layer norm | 26.7 | 55.0 | 35.6 | 45.5 |
| target embedding | 28.0 | 53.8 | 35.9 | 45.2 |
| + layer norm | 28.0 | 53.9 | 35.9 | 45.2 |

the specific source and target sentence, meaning that additional information might provide better translation results. Adding task-specific information to the cross-attention layer has been shown to be helpful. For example, biasing the cross-attention to a monotone alignment provides an improvement in the case of automatic speech recognition [Zhao & Ni$^+$ 20].

Many works on the transformer attention layers investigate strategies to make them more efficient [Tay & Dehghani$^+$ 22]. Most of these approaches consider the self-attention layer and reduce the computation time or memory footprint e.g. by limiting the attention to a selection of the keys [Kitaev & Kaiser$^+$ 20], employing fixed attention patterns [Qiu & Ma$^+$ 20] or simplifying the attention energy computation [Katharopoulos & Vyas$^+$ 20]. However, these strategies cannot be directly applied to cross-attention, and generalized versions do not show the same behavior [Rosendahl & Herold$^+$ 21].

## 5.4 Experimental Evaluation

### 5.4.1 Experimental Setup

The setup for all experiments is identical to the one described in Section 4.4.1. All systems are six-layer transformer architectures trained on English→Italian, Romanian→English German→English or Chinese→English. However, for the experiments described in this chapter no monolingual data is used.

### 5.4.2 Results

#### Word Embeddings

In Section 5.2.1 we describe how the key, value and query vector of the cross-attention layer can be extended by concatenating a word embedding vector. In Table 5.1 we show results for the English→Italian and Romanian→English tasks. First, we observe that concatenating neither the source nor the target word embedding improves the translation performance. In the case of English→Italian we even report a significant loss in performance when providing a source embedding directly to the cross-attention layer. Secondly, we see that the difference between providing the raw embedding vector and its layer-normalized version is negligible. Since layer normalization does not seem to have a positive effect, we do not use when adding word embeddings it in all further experiments.

Providing word embeddings directly to the attention layer does not improve the performance on either low-resource task. From Table 5.2 we conclude that this is a general trend across all four language pairs. The only major performance difference is a degradation of 1.1 BLEU [%] and 1.2 TER [%] when concatenating the source embeddings to encoder output in the English→Italian

Table 5.2: Effect of concatenating word embeddings to the encoder output $h_j$ or the decoder state $s_i$ for all four language pairs. All experiments concatenate raw word embedding, without layer normalization, and all results are reported on the development set.

| | | En→It | | Ro→En | | De→En | | Zh→En | |
|---|---|---|---|---|---|---|---|---|---|
| | | BLEU [%] | TER [%] | BLEU [%] | TER [%] | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| baseline | | 28.1 | 54.0 | 35.7 | 45.3 | 32.7 | 48.6 | 22.2 | 62.4 |
| conc. embd. | src | 27.0 | 55.2 | 35.7 | 45.5 | 32.5 | 48.5 | 22.1 | 62.6 |
| | trg | 28.0 | 53.8 | 35.9 | 45.2 | 32.5 | 48.9 | 22.3 | 62.3 |
| | both | 26.0 | 55.9 | 35.6 | 45.6 | 32.5 | 49.1 | 22.2 | 62.2 |

Table 5.3: Training speed of different attention variations. Training time is given in average hours:minutes per epoch.

| | | En→It | | | Ro→En | | |
|---|---|---|---|---|---|---|---|
| | | train time | | #params | train time | | #params |
| | | h:min | factor | | h:min | factor | |
| baseline | | 0:12 | 1.0 | 52.1M | 0:14 | 1.0 | 54.0M |
| query | target embedding | 0:14 | 1.1 | 53.6M | 0:15 | 1.1 | 55.5M |
| | previous context | 1:54 | 8.7 | 52.3M | 1:37 | 6.9 | 54.1M |
| | accumulated weights | 1:28 | 7.0 | 52.1M | 1:27 | 6.2 | 54.0M |
| key-value | previous context | 1:49 | 8.7 | 55.2M | 1:38 | 6.9 | 57.1M |

setup. Providing a system with both a source and a target embedding leads to degradation over the baseline on three of the four tasks.

We conclude that the encoder outputs $h_1^J$ and decoder states $s_0^I$ are sufficiently good representations of the respective sentences. It is an open debate as to whether the encoder output $h_j$ represents the $j$-th token of the source sentence since the self-attention layers of the encoder have no inherent bias to keep the sequence order. The presented experiments suggest that the encoder output $h_j$ indeed does correspond to the $j$-th source token, since both performance and attention quality do not significantly change when providing the $j$-th word embedding $\tilde{f}_j$. If the word embedding information at position $j$ were contradictory to the output state information $h_j$, we would assume a visible change in the system behavior.

### First-Order Attention Layer

In this section, we present our experimental findings on the higher-order alignment models presented in Section 5.2.2.

**Training time** The first-order extensions of the cross-attention mechanism require the last attention step $i-1$ to be finished before the current step $i$ can be computed. This means that the resulting layer is recurrent rather than feed-forward, as in the vanilla transformer, and cannot be computed for the full target sequence in parallel. This is a major slow-down during training. The training speed for different attention modifications is reported in Table 5.3 and we observe that training takes roughly 7-8 times as long as the baseline training. Note that the increase in training time does not originate from the number of model parameters. The smallest model described in Table 5.3 consists of 52.1M network parameters while the biggest model is only 10% bigger, using 57.1M parameters. This is in clear contrast to the training time differences reported. We observe some variation between these time measurements, which may be caused by external influences such as resource interference with other users in the computing cluster, e.g. high network load.

However, the overall trend is very clear and the lack of decoder parallelization increases training time by a factor of roughly 7×.

While the increased training time is more of an annoyance than a problem for the English→Italian and Romanian→English setups, it means that training a model with higher-order attention on the high-resource tasks German→English and Chinese→English is not feasible. For the case of the German→English task, a baseline system already takes roughly 10-14 days of training time. Hence, a model employing the proposed attention modifications would require more than 70 days of training time or significant GPU parallelization, which is not feasible in the scope of this work. Due to this, we report all results concerning higher-order attention layers only on English→Italian and Romanian→English.

Table 5.4: Translation performance of models with higher-order cross-attention layers. Additional information $\gamma_i$ is concatenated to the existing query vector as described in Equation 5.4. Results are reported on the development set of the respective tasks.

| | | $\gamma_{i,j}$ | En→It | | Ro→En | |
|---|---|---|---|---|---|---|
| | | | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| baseline | | - | 28.1 | 54.0 | 35.7 | 45.3 |
| previous context vector | head specific | $c_{i-1}^{(\ell,m)}$ | 28.3 | 53.8 | 36.3 | 45.2 |
| | full | $c_{i-1}^{(\text{full})}$ | 28.4 | 53.9 | 35.8 | 45.4 |
| previous weight | | $\alpha_{i-1,j}$ | 28.3 | 53.9 | 36.0 | 45.3 |
| accumulated attention weights | | $\sum_{k=1}^{i-1} a_{k,j}$ | 28.1 | 53.9 | 36.0 | 45.1 |
| + fertility | | $\frac{1}{\varphi_j} \sum_{k=1}^{i-1} a_{k,j}$ | 28.2 | 53.8 | 36.2 | 45.1 |

In Table 5.4 we show the impact of higher-order attention layers on the English→Italian and Romanian→English tasks. Providing the head-specific context vector $c_{i-1}^{(\ell,m)}$ of the previous decoding step improves the English→Italian system by 0.2 Bleu [%] and Ter [%] and the Romanian→English system by 0.6 Bleu [%] and 0.1 Ter [%]. Overall, this is a very marginal improvement. Similarly, providing additional information in the form of previous attention weights $\alpha_{i-1}$ does not yield a significant benefit across tasks and metrics. The addition of the accumulated attention weights with fertility yields one of the strongest results, with an average improvement of only 0.3 Bleu [%] and 0.15 Ter [%]. Overall, the higher-order attention layer does not significantly change the performance on the English→Italian task with respect to Bleu or Ter. On the Romanian→English task, we mostly observe small but consistent improvements of 0.3-0.6 Bleu [%], although these are not reflected in the Ter metric. We conclude that a zero-order or feed-forward attention system is strong enough to model the target-to-source alignment in transformer architectures. This is a clear difference from recurrent translation models, where providing fertility information to the cross-attention layer could boost the translation performance by up to 2.0 Bleu [%] [Feng & Liu$^+$ 16].

**Extended key-value list** Next we consider the extended key-value list approach that incorporates past attention decisions by concatenating the previous context vector $c_{i-1}$ to the end of the key-value list. As can be seen from Table 5.5, the new key-value pair has no significant impact on the translation performance. We observe small improvements of up to 0.1-0.4 Bleu [%] and up to 0.4 Ter [%], both of which can be considered noise.

All observations so far are made on the development set. This dataset is involved in the training process of the translation models in two ways: (1) the learning rate is reduced if the loss function of the model does not improve for several consecutive checkpoints, and (2) the final

Table 5.5: Effect of adding the previous context vector $c_{i-1}$ as a new entry pair to the key-value list. Results are reported on the development set of the respective tasks.

| | | En→It | | Ro→En | |
|---|---|---|---|---|---|
| | | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| baseline | | 28.1 | 54.0 | 35.7 | 45.3 |
| prev. context as key-value | head specific | 28.2 | 53.8 | 36.1 | 45.2 |
| | full | 28.2 | 53.6 | 36.0 | 45.3 |

Table 5.6: Final performance of three attention modifications on the test set of the respective tasks. 'TIME ' indicates that the model could not be trained in a reasonable amount of time.

| | | En→It | | Ro→En | | De→En | |
|---|---|---|---|---|---|---|---|
| | | `tst2010` | | `newstest2016` | | `newstest2017` | |
| | | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| baseline | | 29.6 | 52.0 | 34.1 | 47.4 | 33.6 | 48.0 |
| modify query | target embedding | 29.8 | 51.9 | 34.2 | 47.3 | 33.3 | 48.3 |
| | fertility | 30.0 | 51.6 | 34.4 | 47.1 | TIME | TIME |
| extend key-value list | | 29.9 | 51.9 | 34.4 | 47.2 | TIME | TIME |

training checkpoint is selected based on the Bleu performance on the development set. As a result of this, the model could overfit to the development data. In Table 5.6 we report the results of the best-performing methods on the test set of three translation tasks. Concatenating the target embedding to the query is the only attention modification that can be run on all three translation tasks. Similar to the development set performance, it does not show improvements across tasks. On the English→Italian task, we observe an improvement of 0.4 Bleu [%] and Ter [%] when extending the query with accumulated attention weight and fertility. The same method improves the performance on the Romanian→English task by 0.3 Bleu [%] and Ter [%]. Overall, these are very small improvements in the Bleu and Ter scores that can be considered to be noise. We conclude that none of the presented modifications to the attention mechanism significantly improves the translation performance.

### 5.4.3 Comparison to other Work

In this section, we compare our best models to work described in the literature. We use the same benchmark tasks as before, namely the IWSLT 2017 English→Italian, the WMT 2016 Romanian→English and the WMT 2018 German→English task.

The English→Italian task was originally multilingual, providing 20 corpora for all language pairs from the five languages English, German, Dutch, Italian, Romanian, thus many submissions on the task use additional multilingual data. However, all systems reported are comparable in the sense that they use the same English→Italian data. From Table 5.7 we observe that our baseline is already quite strong. On `tst2010` both higher-order attention modifications perform on par with our baseline system, and on `tst2017` they obtain an improvement of 0.6 Bleu [%] over the baseline and the best competing system. This improvement is not reflected in the Ter score of the model using fertility in the attention layer, but concatenating the previous context vector as a key-value pair yields an improvement of 0.8 Bleu [%] and 0.6 Ter [%].

The results for the Romanian→English task are presented in Table 5.8. Our baseline performs comparably to the strongest baseline from [Kasai & Cross$^{+}$ 20], missing only 0.3 Bleu [%]. As observed before, none of the systems with modified attention significantly outperforms a state-of-

Table 5.7: Results of the English→Italian IWSLT task. Systems with multilingual data use more bilingual training data but no additional En→It data. The IWSLT 2017 task allows En ↔ {De,Nl,Ro} data.

| | multilingual | `tst2010` | | `tst2017` | |
|---|---|---|---|---|---|
| | data | BLEU [%] | TER [%] | BLEU [%] | TER [%] |
| [Lakew & Lotito+ 17a] | yes | 28.5 | | | |
| Kyoto IWSLT [Dabre & Cromierès+ 17] | yes | 29.1 | | 30.8 | 50.5 |
| FBK IWSLT [Lakew & Lotito+ 17b] | no | | | 29.9 | |
| + Multilingual | yes | | | 29.6 | 50.7 |
| KIT IWSLT [Pham & Sperber+ 17] | yes | | | 32.0 | 48.4 |
| this work (baseline) | no | 29.6 | 52.0 | 31.9 | 48.1 |
| + trg embedding | | 29.8 | 51.9 | 32.0 | 47.9 |
| + fertility | | 30.0 | 51.6 | 32.5 | 47.9 |
| + prev. context as key-value | | 29.9 | 51.9 | 32.6 | 47.5 |

Table 5.8: Romanian→English: Final results on attention extensions in comparison to the results from other works.

| model | `newstest2016` | |
|---|---|---|
| | BLEU [%] | TER [%] |
| RNN attention [Sennrich & Haddow+ 16a] | 29.2 | |
| transformer [Kasai & Cross+ 20] | 34.5 | |
| this work (baseline) | 34.2 | 47.4 |
| + trg embedding | 34.2 | 47.3 |
| + fertility | 34.4 | 47.1 |
| + prev. context as key-value | 34.4 | 47.2 |

the-art baseline.

On the German→English tasks we are restricted to maintaining the parallelization during training of the transformer architecture. Hence, we only report the experiments where the target embedding is concatenated to the query. From the results in Table 5.9 we confirm that modifying the attention queries by providing target embeddings does not help to strengthen the system. Furthermore, we observe that other works report no improvements when using a first-order neural alignment model within an HMM approach modeled with a transformer architecture [Wang 23]. This further indicates that additional alignment information over the decoder steps $i$ is not beneficial.

## 5.5 Summary and Conclusion

In this chapter, we investigated and extended the cross-attention layer in transformer machine translation models. We considered the attention weights as an implicit, target-to-source, soft alignment. In contrast to explicit, count-based alignment models [Vogel & Ney+ 96, Och & Ney 03], the cross-attention layer operates as a zero-order model. We investigated several extensions to the cross-attention layer of the transformer which provide information from the previous decoder steps $i' < i$ to the attention weight calculation. The investigation focused on two questions:

(a) Are higher-order attention models beneficial?

(b) How to represent and incorporate additional information into the attention mechanism?

Table 5.9: German→English: Final results on attention extensions in comparison to the results from other works.

| | dev (newstest2015) | | `newstest2017` | |
|---|---|---|---|---|
| | Bleu [%] | Ter [%] | Bleu [%] | Ter [%] |
| zero-order direct HMM [Wang 23] | 32.3 | 49.5 | 33.8 | 49.5 |
| first-order direct HMM [Wang 23] | 32.5 | 49.2 | 33.8 | 49.4 |
| zero-order direct HMM [Bahar 22] | | | 33.7 | |
| transformer [Tang & Müller$^+$ 18] | | | 33.7 | |
| convolution [Tang & Müller$^+$ 18] | | | 30.4 | |
| baseline | 32.7 | 48.6 | 33.6 | 48.0 |
| + trg embeddings | 32.5 | 49.1 | 33.1 | 48.7 |

These two questions are directly connected. Since the baseline cross-attention layer is a zero-order model, each answer to question (a) directly requires an answer to question (b).

Building upon existing works [Feng & Liu$^+$ 16, Tu & Lu$^+$ 16], we extended the attention layer by providing token embeddings, attention weights, context vectors or accumulated energies directly to the attention layer. To incorporate this new information we interpreted the attention layer as a soft lookup of a query on a sequence of key-value pairs. We modified each of these components and investigated where the additional information is processed best by the model.

Throughout our experiments, we observed a small improvement of 0.2-0.6 Bleu [%] across languages when extending the query with higher-order dependencies (Table 5.4). While having an overall positive trendd this improvement is minor and we consider it to be noise. We observed no performance distinction between first- and higher-order cross-attention layers; however, in our experiments the higher-order representations are severely compressed along the decoder time axis.

We proposed an approach to extend the key-value sequence of the cross-attention layer to incorporate previous attention states as first-order dependencies. The resulting model performs 0.3 Bleu [%] and around 0.1 Ter [%] better than the baseline model (Table 5.6).

Throughout this chapter, we have considered different representations of past attention decisions and different ways to incorporate them into the cross-attention layer. Whether these can be beneficial is part of an ongoing scientific discussion, where some groups have reported strong improvements in translation quality [Feng & Liu$^+$ 16, Tu & Lu$^+$ 16] while others report no improvements [Peter 20]. We contributed to the debate by proposing new approaches with first-order dependencies and adapting existing extensions to the transformer architecture. This is an important distinction from previous works, which focused on recurrent machine translation systems with one single-headed, additive cross-attention layer. Testing the described extensions in a state-of-the-art architecture, we concluded that higher-order dependencies do not benefit the cross-attention layer in a transformer model.

Overall, this brings us to the conclusion that the cross-attention layer in transformer models obtains enough context information and that it does not require a first-order extension.

## 5.6 Individual Contributions

In this section, we list the individual contributions of the author in contrast to the work of colleagues in joint research projects related to this dissertation. In particular, in accordance with §5.6 of the doctoral guidelines of the RWTH Aachen University, we note which publications of the author overlap with the presented work.

The implementation of the attention modifications was done by Frithjof Petrick during his bachelor thesis [Petrick 20], which was closely supervised by the author and Christian Herold.

The original ideas were developed by the author and Christian Herold in direct discussions and are based on similar work on recurrent networks [Feng & Liu$^+$ 16, Peter 20]. Refinements of the ideas and modeling decisions are a product of all three contributors (Jan Rosendahl, Christian Herold, Frithjof Petrick) throughout weekly meetings and discussions. The results are also published at an ACL 2021 workshop [Rosendahl & Herold$^+$ 21]. All experiments in this thesis are performed by the author unless explicitly stated at the experiments (e.g. when comparing to work from the literature).

# 6. Scientific Achievements

In this work, we investigate two important components of a machine translation system: the training data and the cross-attention layer. Namely, we increase the amount of data available by incorporating monolingual data (Chapter 4) and extend the encoder-decoder cross-attention to incorporate higher-order dependencies (Chapter 5). The goals for each approach is outlined in Chapter 2 and we discuss the findings for both approaches individually.

## Monolingual Data in Machine Translation

- We introduce a fusion model, which is a log-linear combination of a neural translation and language model. This is an adaption of count-based translation systems, which often improves the translation performance when a language model is used. We compare a symbol- and sequence-level normalization approach and introduce a new algorithm to train the translation model as part of the full fusion model. We observe improvements when integrating a language model into the decision rule (Tables 4.11-4.12), which depend heavily on the domain of the training, testing and monolingual data (Table 4.16).
- To calculate the training criterion of a sequence-level normalized fusion model we present and investigate different approximation strategies based on beam search and context-reduced models. Training of the translation model as part of a sequence-level normalized fusion model does not provide performance improvements (Table 4.18) and shows problematic convergence behavior in the desired metrics (Figure 4.8). This could be caused by either a mismatch between the training criterion and the evaluation metrics Bleu and Ter or an insufficient approximation of the training criterion, which requires an approximation of the sum over all target sentences.
  While these approaches fail to work for machine translation, the idea and implementation are used in automatic speech recognition [Wynands & Michel$^+$ 22].
- We introduce a pre-training method that uses monolingual data to select a better initialization point for the main training on bilingual data. Using a language model or cloze loss allows the pre-training of different parts of the machine translation system. We observe strong performance improvements on low-resource tasks (Table 4.26) and smaller impacts on high-resource tasks (Table 4.27).
- We extend the pre-training approach to a multi-task training that optimizes several loss functions in parallel. We show that this results in worse performance than pre-training (Table 4.32).
- Pre-training introduces a new way of using monolingual source data and we observe great performance improvements for low-resource tasks (Table 4.21). We show that these improvements can be attributed equally to the impact of the source data and a direct loss on the encoder.

- We verify the performance of back-translation, the strongest and most common approach to including monolingual target data. Our findings are consistent with the literature and we use the back-translation systems as a strong baseline.
- We compare all three methods that use monolingual data on four different translation tasks. We observe that monolingual data is an important resource since all methods improve the performance of the machine translation system. Overall, back-translation yields the best results for each of the four tasks (Tables 4.32-4.35).
- We analyze the effect of domains and conclude that all approaches are susceptible to domain changes. Including monolingual data is an especially powerful tool if no bilingual training data from the targeted domain is available (Table 4.37).
- We combine our strongest methods and report improvements over a system with back-translation on two of the four tasks (Table 4.39).

### Encoder-Decoder Cross-Attention

- We change the cross-attention layer by extending the dependencies. In particular, we use zero-, first- and higher-order dependencies to model the attention similar to an alignment.
- Building upon existing work [Feng & Liu$^+$ 16, Tu & Lu$^+$ 16], we extend the attention layer by providing previous attention weights, context vectors or accumulated energies to the cross-attention layer, each being a different representation of the previous attention decisions. We observe no significant improvements over the performance of the baseline model (Table 5.4).
- Introducing previous attention decisions as an artificial input token to focus on, we extend the key-value list of the cross-attention layer to include the output of the previous attention decoder step. This allows an explicit attention weight to be put on the first-order dependency. However, the approach shows no improvement compared to the baseline system (Table 5.4).
- Introducing a first-order dependency in the cross-attention layer converts the transformer into a recurrent architecture. We observe that the computational costs in training increase by a factor of seven (Table 5.3). While search is not affected, this means that these approaches are not suitable for large models on high-resource tasks.
- We contribute to the ongoing scientific debate on whether a first-order cross-attention is beneficial, with some groups reporting positive changes [Feng & Liu$^+$ 16, Tu & Lu$^+$ 16] and other work showing no impact [Peter 20]. By proposing new approaches with first-order dependencies and adapting existing extensions to the transformer architecture, we conclude that higher-order dependencies do not benefit the cross-attention layer in a state-of-the-art translation model.

# A. Appendix

## A.1 Corpora

### A.1.1 IWSLT English→Italian

Table A.1 shows the resource provided by the English→Italian language pair of the IWSLT task on multilingual translation [Cettolo & Federico+ 17]. We use only the English→Italian parallel data for all experiments in this work. The data originates from TED talks[1], i.e. scientific talks for a wider audience held in English. We use `dev2010` as the development set and `tst2010` and `tst2017` test sets. To investigate the effects on different domains we use `newstest2009` from the WMT and obtained from the OPUS [Tiedemann 12] website[2].

Table A.1: English→Italian: statistics of the training, evaluation and test data sets (IWSLT shared task).

| | | English | Italian |
|---|---|---|---|
| train | Sentences | 232k | |
| | Running Words | 4.7M | 4.4M |
| | Vocabulary | 70.7k | 103.6k |
| monolingual | Sentences | 92.5M | 46.0M |
| | Running Words | 2.5G | 1.2G |
| dev (dev2010) | Sentences | 929 | |
| | Running Words | 20.2k | 18.1k |
| | OOVs | 235 (1.2%) | 320 (1.8%) |
| tst2010 | Sentences | 1.6k | |
| | Running Words | 31.2k | 28.9k |
| | OOVs | 212 (0.7%) | 293 (1.0%) |
| newstest2009 | Sentences | 3.0k | |
| | Running Words | 77.4k | 76.7k |
| | OOVs | 3.151 (4.1%) | 3.361 (4.4%) |

### A.1.2 WMT 2016 Romanian→English

We run experiments on the Romanian→English task from WMT 2016 [Bojar & Chatterjee+ 16] and show the corpus overview in Table A.2. The bilingual training corpus consists of `Europarl v8` and `SETIMES2`. English monolingual data is obtained from the `NewsCrawl 2016-2018` and Romanian monolingual data from `NewsCrawl 2015`. We use the standard `updated_newsdev2016`

[1] `https://www.ted.com/`
[2] `https://opus.nlpl.eu/WMT-News.php`

as the development set and `newstest2016` as the test set. All data was obtained via the WMT website[3].

Table A.2: Romanian→English: statistics of the training, evaluation and test data sets (WMT 2016 shared task).

| | | Romanian | English |
|---|---|---|---|
| train | Sentences | 612k | |
| | Running Words | 16.2M | 15.9M |
| | Vocabulary | 155.1k | 104.6k |
| monolingual | Sentences | 2.3M | 63.2M |
| | Running Words | 54.8M | 1.4G |
| dev (updated_newsdev2016) | Sentences | 2.0k | |
| | Running Words | 51.7k | 49.9k |
| | OOVs | 2.6k (5.1%) | 1.7k (3.4%) |
| newstest2016 | Sentences | 2.0k | |
| | Running Words | 49.2k | 47.9k |
| | OOVs | 2.2k (4.5%) | 1.4k (2.8%) |

### A.1.3 WMT 2017 German→English

Table A.3: German→English: statistics of the training, evaluation and test data sets (WMT 2017 shared task).

| | | German | English |
|---|---|---|---|
| train: | Sentences | 4.6M | |
| | Running Words | 111.1M | 117.7M |
| | Vocabulary | 2.1M | 1.0M |
| monolingual | Sentences | 142.2M | 45.0M |
| | Running Words | 2.4G | 1.0G |
| dev (newstest2105): | Sentences | 2.2k | |
| | Running Words | 44.1k | 46.8k |
| | OOVs | 1124 (2.6%) | 563 (1.2%) |
| newstest2014: | Sentences | 3.0k | |
| | Running Words | 63.0k | 67.6k |
| | OOVs | 1641 (2.6%) | 824 (1.2%) |
| newstest2017: | Sentences | 3.0k | |
| | Running Words | 61.0k | 64.8k |
| | OOVs | 1593 (2.6%) | 708 (1.0%) |
| newstest2018: | Sentences | 3.0k | |
| | Running Words | 64.0k | 67.5k |
| | OOVs | 1641 (2.6%) | 836 (1.2%) |

For German→English we use the data specified by the WMT 2018 shared task on news translation [Bojar & Federmann+ 18] and give an overview in Table A.3. The training data consists of the `Europarl v7`, `Commoncrawl`, `News Commentary v13` and `Rapid` corpora. Monolingual data in English and German is obtained from the `News Crawl 2017` and `News Crawl 2018` corpora

[3]`https://www.statmt.org/wmt16/translation-task.html`

in the respective languages. We use `newstest2015` as a development set and report results on various `newstest20XX`. All data was downloaded from the WMT website[4].

### A.1.4 WMT 2017 Chinese→English

Table A.4: Chinese→English: statistics of the training, evaluation and test data sets (WMT 2018 shared task).

| | | Chinese | English |
|---|---|---|---|
| train | Sentences | 17.0M | |
| | Running Words | 99.9M | 389.0M |
| | Vocabulary | 25.2M | 0.9M |
| monolingual | Sentences | 14.0M | 45.0M |
| | Running Words | 117.2M | 1.0G |
| dev (newsdev2017) | Sentences | 2.0k | |
| | Running Words | 13.8k | 58.5k |
| | OOVs | 5.7k (41.5 %) | 480 (0.8 %) |
| newstest2017 | Sentences | 2.0k | |
| | Running Words | 14.0k | 54.0k |
| | OOVs | 5.6k (39.7%) | 504k (0.9 %) |
| newstest2018 | Sentences | 4.0k | |
| | Running Words | 27.7k | 101.7k |
| | OOVs | 11.1k (40.2%) | 923 (0.8 %) |

We perform experiments on the Chinese→English shared task on news translation of the WMT 2018 [Bojar & Federmann$^+$ 18]. The parallel training data consists of `News Commentary v13`, `UN Parallel Corpus V1.0` and `CWMT Corpus`. The data is filtered to 17M lines using a unicode-based filtering technique [Gao & Wang$^+$ 19] to be directly comparable to [Bahar 22, Wang 23]. We obtain English monolingual data from the `News Crawl2017` and `News Crawl2018` and Chinese data from `News Crawl2010-2021`. All data was downloaded from the WMT website[5].

[4]`https://www.statmt.org/wmt18/translation-task.html`
[5]`https://www.statmt.org/wmt18/translation-task.html`

# List of Figures

# List of Tables

# Bibliography

[Abadi & Agarwal$^+$ 15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[Aharoni & Johnson$^+$ 19] R. Aharoni, M. Johnson, O. Firat: Massively Multilingual Neural Machine Translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3874–3884, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[Akhbardeh & Arkhangorodsky$^+$ 21] F. Akhbardeh, A. Arkhangorodsky, M. Biesialska, O. Bojar, R. Chatterjee, V. Chaudhary, M.R. Costa-jussa, C. España-Bonet, A. Fan, C. Federmann, M. Freitag, Y. Graham, R. Grundkiewicz, B. Haddow, L. Harter, K. Heafield, C. Homan, M. Huck, K. Amponsah-Kaakyire, J. Kasai, D. Khashabi, K. Knight, T. Kocmi, P. Koehn, N. Lourie, C. Monz, M. Morishita, M. Nagata, A. Nagesh, T. Nakazawa, M. Negri, S. Pal, A.A. Tapo, M. Turchi, V. Vydrin, M. Zampieri: Findings of the 2021 Conference on Machine Translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, pp. 1–88, Online, Nov. 2021. Association for Computational Linguistics.

[Al-Rfou & Choe$^+$ 19] R. Al-Rfou, D. Choe, N. Constant, M. Guo, L. Jones: Character-Level Language Modeling with Deeper Self-Attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, No. 01, pp. 3159–3166, Jul. 2019.

[Alkhouli & Bretschner$^+$ 16] T. Alkhouli, G. Bretschner, J.T. Peter, M. Hethnawi, A. Guta, H. Ney: Alignment-Based Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pp. 54–65, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[Alkhouli & Bretschner$^+$ 18] T. Alkhouli, G. Bretschner, H. Ney: On The Alignment Problem In Multi-Head Attention-Based Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 177–185, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.

[Alkhouli & Ney 17] T. Alkhouli, H. Ney: Biasing Attention-Based Recurrent Neural Networks Using External Alignment Information. In *Proceedings of the Second Conference on Machine*

*Translation*, pp. 108–117, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[Andor & Alberti$^+$ 16] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, M. Collins: Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, August 2016.

[Artetxe & Labaka$^+$ 18] M. Artetxe, G. Labaka, E. Agirre, K. Cho: Unsupervised Neural Machine Translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, April 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*, pp. 194–203, Florence, Italy, July 2019. Association for Computational Linguistics.

[Ba & Kiros$^+$ 16] J.L. Ba, J.R. Kiros, G.E. Hinton: Layer normalization, Juli 2016.

[Bahar 22] P. Bahar: *Neural Sequence-to-Sequence Modeling for Language and Speech Translation*. Ph.D. thesis, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, Nov. 2022.

[Bahar & Alkhouli$^+$ 17] P. Bahar, T. Alkhouli, J.T. Peter, C.J.S. Brix, H. Ney: Empirical Investigation of Optimization Algorithms in Neural Machine Translation. In *Conference of the European Association for Machine Translation*, pp. 13–26, Prague, Czech Republic, June 2017.

[Bahar & Rosendahl$^+$ 17] P. Bahar, J. Rosendahl, N. Rossenbach, H. Ney: The RWTH Aachen Machine Translation Systems for IWSLT 2017. In *International Workshop on Spoken Language Translation*, pp. 29–34, Tokyo, Japan, Dec. 2017.

[Bahdanau & Cho$^+$ 15] D. Bahdanau, K. Cho, Y. Bengio: Neural Machine Translation by Jointly Learning to Align and Translate. In Y. Bengio, Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, Conference Track Proceedings*, May 2015.

[Bañón & Chen$^+$ 20] M. Bañón, P. Chen, B. Haddow, K. Heafield, H. Hoang, M. Esplà-Gomis, M.L. Forcada, A. Kamran, F. Kirefu, P. Koehn, S. Ortiz Rojas, L. Pla Sempere, G. Ramírez-Sánchez, E. Sarrías, M. Strelec, B. Thompson, W. Waites, D. Wiggins, J. Zaragoza: ParaCrawl: Web-Scale Acquisition of Parallel Corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4555–4567, Online, July 2020. Association for Computational Linguistics.

[Barrault & Biesialska$^+$ 20] L. Barrault, M. Biesialska, O. Bojar, M.R. Costa-jussà, C. Federmann, Y. Graham, R. Grundkiewicz, B. Haddow, M. Huck, E. Joanis, T. Kocmi, P. Koehn, C.k. Lo, N. Ljubešić, C. Monz, M. Morishita, M. Nagata, T. Nakazawa, S. Pal, M. Post, M. Zampieri: Findings of the 2020 Conference on Machine Translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pp. 1–55, Online, Nov. 2020. Association for Computational Linguistics.

[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 O. Bojar,

R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, A. Martins, C. Monz, M. Negri, A. Névéol, M.L. Neves, M. Post, M. Turchi, K. Verspoor, editors, *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 2: Shared Task Papers, Day 1*, pp. 1–61. Association for Computational Linguistics, August 2019.

[Bertoldi & Federico 09] N. Bertoldi, M. Federico: Domain Adaptation for Statistical Machine Translation with Monolingual Resources. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pp. 182–189, Athens, Greece, March 2009. Association for Computational Linguistics.

[Bojar & Buck$^+$ 14] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, A. Tamchyna: Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

[Bojar & Chatterjee$^+$ 16] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Névéol, M.L. Neves, M. Popel, M. Post, R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor, M. Zampieri: Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany*, pp. 131–198. The Association for Computer Linguistics, August 2016.

[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 O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, C. Monz, M. Negri, A. Névéol, M.L. Neves, M. Post, L. Specia, M. Turchi, K. Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Shared Task Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pp. 272–303. Association for Computational Linguistics, October 2018.

[Bojar & Tamchyna 11] O. Bojar, A. Tamchyna: Improving Translation Model by Monolingual Data. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pp. 330–336, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.

[Brants & Popat$^+$ 07] T. Brants, A.C. Popat, P. Xu, F.J. Och, J. Dean: Large Language Models in Machine Translation. In J. Eisner, editor, *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pp. 858–867. ACL, June 2007.

[Britz & Goldie$^+$ 17] D. Britz, A. Goldie, M.T. Luong, Q. Le: Massive Exploration of Neural Machine Translation Architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1442–1451, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[Brown & Cocke$^+$ 90] P.F. Brown, J. Cocke, S.D. Pietra, V.J.D. Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, P.S. Roossin: A Statistical Approach to Machine Translation. *International Conference on Computational Linguistics*, Vol. 16, No. 2, pp. 79–85, 1990.

[Brown & Mann$^+$ 20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger,

T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei: Language Models are Few-Shot Learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin, editors, *Advances in Neural Information Processing Systems*, Vol. 33, pp. 1877–1901. Curran Associates, Inc., 2020.

[Brown & Pietra$^+$ 93] P.F. Brown, S.D. Pietra, V.J.D. Pietra, R.L. Mercer: The Mathematics of Statistical Machine Translation: Parameter Estimation. *Comput. Linguistics*, Vol. 19, No. 2, pp. 263–311, 1993.

[Callison-Burch & Osborne$^+$ 06] C. Callison-Burch, M. Osborne, P. Koehn: Re-evaluating the Role of Bleu in Machine Translation Research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 249–256, Trento, Italy, April 2006. Association for Computational Linguistics.

[Carrión-Ponz & Casacuberta 22] S. Carrión-Ponz, F. Casacuberta: On the Effectiveness of Quasi Character-Level Models for Machine Translation. In *Proceedings of the 15th biennial conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pp. 131–143, Orlando, USA, Sept. 2022. Association for Machine Translation in the Americas.

[Caruana 93] R. Caruana: Multitask Learning: A Knowledge-Based Source of Inductive Bias. In P.E. Utgoff, editor, *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, pp. 41–48. Morgan Kaufmann, June 1993.

[Caswell & Chelba$^+$ 19] I. Caswell, C. Chelba, D. Grangier: Tagged Back-Translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pp. 53–63, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[Cettolo & Federico$^+$ 17] M. Cettolo, M. Federico, L. Bentivogli, J. Niehues, S. Stüker, K. Sudoh, K. Yoshino, C. Federmann: Overview of the IWSLT 2017 Evaluation Campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pp. 2–14, Tokyo, Japan, Dec. 14-15 2017. International Workshop on Spoken Language Translation.

[Cettolo & Girardi$^+$ 12] M. Cettolo, C. Girardi, M. Federico: WIT3: Web Inventory of Transcribed and Translated Talks. In *Proceedings of the 16th Annual conference of the European Association for Machine Translation*, pp. 261–268, Trento, Italy, May 28–30 2012. European Association for Machine Translation.

[Chaudhary & Tang$^+$ 19] V. Chaudhary, Y. Tang, F. Guzmán, H. Schwenk, P. Koehn: Low-Resource Corpus Filtering Using Multilingual Sentence Embeddings. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pp. 261–266, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[Chen & Firat$^+$ 18] M.X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, M. Schuster, N. Shazeer, N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, Z. Chen, Y. Wu, M. Hughes: The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–86, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[Cho & van Merrienboer$^+$ 14] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In A. Moschitti, B. Pang, W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734. ACL, October 2014.

[Clark & Luong$^+$ 20] K. Clark, M. Luong, Q.V. Le, C.D. Manning: ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, April 2020.

[Cohn & Hoang$^+$ 16] T. Cohn, C.D.V. Hoang, E. Vymolova, K. Yao, C. Dyer, G. Haffari: Incorporating Structural Alignment Biases into an Attentional Neural Translation Model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 876–885, San Diego, California, June 2016. Association for Computational Linguistics.

[Collobert & Weston 08] R. Collobert, J. Weston: A unified architecture for natural language processing: deep neural networks with multitask learning. In W.W. Cohen, A. McCallum, S.T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, Vol. 307 of *ACM International Conference Proceeding Series*, pp. 160–167. ACM, June 2008.

[Collobert & Weston$^+$ 11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P.P. Kuksa: Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.*, Vol. 12, pp. 2493–2537, 2011.

[Conneau & Lample 19] A. Conneau, G. Lample: Cross-lingual Language Model Pretraining. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett, editors, *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc., December 2019.

[Costa-jussà & Escolano$^+$ 17] M.R. Costa-jussà, C. Escolano, J.A.R. Fonollosa: Byte-based Neural Machine Translation. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pp. 154–158, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[Costa-jussà & Fonollosa 16] M.R. Costa-jussà, J.A.R. Fonollosa: Character-based Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 357–361, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[Currey & Miceli Barone$^+$ 17] A. Currey, A.V. Miceli Barone, K. Heafield: Copied Monolingual Data Improves Low-Resource Neural Machine Translation. In *Proceedings of the Second Conference on Machine Translation*, pp. 148–156, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[Dabre & Cromierès$^+$ 17] R. Dabre, F. Cromierès, S. Kurohashi: Kyoto University MT System Description for IWSLT 2017. In S. Sakti, M. Utiyama, editors, *Proceedings of the 14th International Conference on Spoken Language Translation, IWSLT 2017, Tokyo, Japan, December 14-15, 2017*, pp. 55–59. International Workshop on Spoken Language Translation, December 2017.

[Dai & Yang$^+$ 19] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, R. Salakhutdinov: Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[Devlin & Chang$^+$ 19] J. Devlin, M. Chang, K. Lee, K. Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran, T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, June 2019.

[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 *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 5345–5349, New Orleans, LA, USA, March 2017.

[Dou & Yu$^+$ 19] Z. Dou, K. Yu, A. Anastasopoulos: Investigating Meta-Learning Algorithms for Low-Resource Natural Language Understanding Tasks. In K. Inui, J. Jiang, V. Ng, X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 1192–1197. Association for Computational Linguistics, 2019.

[Edunov & Ott$^+$ 18] S. Edunov, M. Ott, M. Auli, D. Grangier: Understanding Back-Translation at Scale. In E. Riloff, D. Chiang, J. Hockenmaier, J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 489–500. Association for Computational Linguistics, October 2018.

[Fadaee & Bisazza$^+$ 17] M. Fadaee, A. Bisazza, C. Monz: Data Augmentation for Low-Resource Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 567–573, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[Fan & Gong$^+$ 21] Z. Fan, Y. Gong, D. Liu, Z. Wei, S. Wang, J. Jiao, N. Duan, R. Zhang, X. Huang: Mask Attention Networks: Rethinking and Strengthen Transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1692–1701, Online, June 2021. Association for Computational Linguistics.

[Feng & Liu$^+$ 16] S. Feng, S. Liu, N. Yang, M. Li, M. Zhou, K.Q. Zhu: Improving Attention Modeling with Implicit Distortion and Fertility for Machine Translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 3082–3092, Osaka, Japan, Dec. 2016. The COLING 2016 Organizing Committee.

[Freitag & Grangier$^+$ 20] M. Freitag, D. Grangier, I. Caswell: BLEU might be Guilty but References are not Innocent. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 61–71, Online, Nov. 2020. Association for Computational Linguistics.

[Freitag & Rei$^+$ 21] M. Freitag, R. Rei, N. Mathur, C.k. Lo, C. Stewart, G. Foster, A. Lavie, O. Bojar: Results of the WMT21 Metrics Shared Task: Evaluating Metrics with Expert-based Human Evaluations on TED and News Domain. In *Proceedings of the Sixth Conference on Machine Translation*, pp. 733–774, Online, Nov. 2021. Association for Computational Linguistics.

[Gao & Wang$^+$ 19] Y. Gao, W. Wang, H. Ney: uniblock: Scoring and Filtering Corpus with Unicode Block Information. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1324–1329, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[Gao & Wang$^+$ 20] Y. Gao, W. Wang, C. Herold, Z. Yang, H. Ney: Towards a Better Understanding of Label Smoothing in Neural Machine Translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pp. 212–223, Suzhou, China, Dec. 2020. Association for Computational Linguistics.

[Gardner & Grus$^+$ 18] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N.F. Liu, M. Peters, M. Schmitz, L. Zettlemoyer: AllenNLP: A Deep Semantic Natural Language Processing Platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pp. 1–6, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[Garg & Peitz$^+$ 19] S. Garg, S. Peitz, U. Nallasamy, M. Paulik: Jointly Learning to Align and Translate with Transformer Models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4453–4462, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[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 (Volume 1: Research Papers)*, pp. 45–52, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[Gülçehre & Firat$^+$ 15] Ç. Gülçehre, O. Firat, K. Xu, K. Cho, L. Barrault, H. Lin, F. Bougares, H. Schwenk, Y. Bengio: On Using Monolingual Corpora in Neural Machine Translation. *CoRR*, Vol. abs/1503.03535, 2015.

[Gülçehre & Firat$^+$ 17] Ç. Gülçehre, O. Firat, K. Xu, K. Cho, Y. Bengio: On integrating a language model into neural machine translation. *Comput. Speech Lang.*, Vol. 45, pp. 137–148, 2017.

[Ha & Niehues$^+$ 16] T.L. Ha, J. Niehues, A. Waibel: Toward Multilingual Neural Machine Translation with Universal Encoder and Decoder. In *Proceedings of the 13th International Conference on Spoken Language Translation*, Seattle, Washington D.C, Dec. 8-9 2016. International Workshop on Spoken Language Translation.

[Haddow & Bawden$^+$ 22] B. Haddow, R. Bawden, A.V. Miceli Barone, J. Helcl, A. Birch: Survey of Low-Resource Machine Translation, 06 2022.

[He & Xia$^+$ 16] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T. Liu, W. Ma: Dual Learning for Machine Translation. In D.D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 820–828, December 2016.

[He & Zhang$^+$ 16] K. He, X. Zhang, S. Ren, J. Sun: Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, June 2016.

[Herold & Rosendahl$^+$ 21] C. Herold, J. Rosendahl, J. Vanvinckenroye, H. Ney: Data Filtering using Cross-Lingual Word Embeddings. In *2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, online, May 2021.

[Herold & Rosendahl$^+$ 22] C. Herold, J. Rosendahl, J. Vanvinckenroye, H. Ney: Detecting Various Types of Noise for Neural Machine Translation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2542–2551, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[Hill & Cho$^+$ 15] F. Hill, K. Cho, S. Jean, C. Devin, Y. Bengio: Embedding Word Similarity with Neural Machine Translation. In Y. Bengio, Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, May 2015.

[Hochreiter & Schmidhuber 97] S. Hochreiter, J. Schmidhuber: Long Short-Term Memory. *Neural Comput.*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[Huang & Pérez$^+$ 20] X.S. Huang, F. Pérez, J. Ba, M. Volkovs: Improving Transformer Optimization Through Better Initialization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Vol. 119 of *Proceedings of Machine Learning Research*, pp. 4475–4483. PMLR, July 2020.

[Huang & Xu$^+$ 15] Z. Huang, W. Xu, K. Yu: Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR*, Vol. abs/1508.01991, 2015.

[Huck & Riess$^+$ 17] M. Huck, S. Riess, A. Fraser: Target-side Word Segmentation Strategies for Neural Machine Translation. In *Proceedings of the Second Conference on Machine Translation*, pp. 56–67, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[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 2nd Workshop on Neural Machine Translation and Generation*, pp. 55–63, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[Irie & Zeyer$^+$ 19] K. Irie, A. Zeyer, R. Schlüter, H. Ney: Language Modeling with Deep Transformers. In *Interspeech*, pp. 3905–3909, Graz, Austria, Sept. 2019. ISCA Best Student Paper Award. [slides].

[Johnson & Schuster$^+$ 17] M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, J. Dean: Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguistics*, Vol. 5, pp. 339–351, 2017.

[Jon & Popel$^+$ 22] J. Jon, M. Popel, O. Bojar: CUNI-Bergamot Submission at WMT22 General Translation Task. In *Proceedings of the Seventh Conference on Machine Translation*, pp. 280–289, Abu Dhabi, December 2022. Association for Computational Linguistics.

[Joshi & Chen$^+$ 20] M. Joshi, D. Chen, Y. Liu, D.S. Weld, L. Zettlemoyer, O. Levy: SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics*, Vol. 8, pp. 64–77, 2020.

[Kasai & Cross$^+$ 20] J. Kasai, J. Cross, M. Ghazvininejad, J. Gu: Non-autoregressive Machine Translation with Disentangled Context Transformer. In *Proceedings of the 37th International*

*Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Vol. 119 of *Proceedings of Machine Learning Research*, pp. 5144–5155. PMLR, July 2020.

[Kasai & Peng$^+$ 21] J. Kasai, H. Peng, Y. Zhang, D. Yogatama, G. Ilharco, N. Pappas, Y. Mao, W. Chen, N.A. Smith: Finetuning Pretrained Transformers into RNNs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10630–10643, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[Katharopoulos & Vyas$^+$ 20] A. Katharopoulos, A. Vyas, N. Pappas, F. Fleuret: Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Vol. 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, July 2020.

[Kim 22] Y. Kim: *Neural Machine Translation for Low-Resource Scenarios*. Ph.D. thesis, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, Feb. 2022.

[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*, pp. 1246–1257, Florence, Italy, July 2019. Association for Computational Linguistics.

[Kim & Graça$^+$ 20] Y. Kim, M. Graça, H. Ney: When and Why is Unsupervised Neural Machine Translation Useless? In M.L. Forcada, A. Martins, H. Moniz, M. Turchi, A. Bisazza, J. Moorkens, A.G. Arenas, M. Nurminen, L. Marg, S. Fumega, B. Martins, F. Batista, L. Coheur, C.P. Escartín, I. Trancoso, editors, *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, EAMT 2020, Lisboa, Portugal, November 3-5, 2020*, pp. 35–44. European Association for Machine Translation, November 2020.

[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 *ACL Workshop on Representation Learning for NLP*, Florence, Italy, July 2019. [poster].

[Kingma & Ba 15] D.P. Kingma, J. Ba: Adam: A Method for Stochastic Optimization. In Y. Bengio, Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Kitaev & Kaiser$^+$ 20] N. Kitaev, L. Kaiser, A. Levskaya: Reformer: The Efficient Transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, April 2020.

[Kneser & Ney 95] R. Kneser, H. Ney: Improved backing-off for M-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '95, Detroit, Michigan, USA, May 08-12, 1995*, pp. 181–184. IEEE Computer Society, May 1995.

[Kocmi & Bawden$^+$ 22] T. Kocmi, R. Bawden, O. Bojar, A. Dvorkovich, C. Federmann, M. Fishel, T. Gowda, Y. Graham, R. Grundkiewicz, B. Haddow, R. Knowles, P. Koehn, C. Monz, M. Morishita, M. Nagata, T. Nakazawa, M. NovÃ¡k, M. Popel, M. PopoviÄ‡, M. Shmatova: Findings of the 2022 Conference on Machine Translation (WMT22). In *Proceedings of the Seventh Conference on Machine Translation*, pp. 1–45, Abu Dhabi, December 2022. Association for Computational Linguistics.

[Kocmi & Bojar 18] T. Kocmi, O. Bojar: Trivial Transfer Learning for Low-Resource Neural Machine Translation. In O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, C. Monz, M. Negri, A. Névéol, M.L. Neves, M. Post, L. Specia, M. Turchi, K. Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pp. 244–252. Association for Computational Linguistics, October 2018.

[Koehn 10] P. Koehn: *Statistical Machine Translation.* Cambridge University Press, 2010.

[Koehn & Chaudhary$^+$ 20] P. Koehn, V. Chaudhary, A. El-Kishky, N. Goyal, P.J. Chen, F. Guzmán: Findings of the WMT 2020 Shared Task on Parallel Corpus Filtering and Alignment. In *Proceedings of the Fifth Conference on Machine Translation*, pp. 726–742, Online, Nov. 2020. Association for Computational Linguistics.

[Koehn & Hoang$^+$ 07] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst: Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pp. 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Koehn & Knowles 17] P. Koehn, R. Knowles: Six Challenges for Neural Machine Translation. In T. Luong, A. Birch, G. Neubig, A.M. Finch, editors, *Proceedings of the First Workshop on Neural Machine Translation, NMT@ACL 2017, Vancouver, Canada, August 4, 2017*, pp. 28–39. Association for Computational Linguistics, 2017.

[Koehn & Monz 06] P. Koehn, C. Monz: Manual and Automatic Evaluation of Machine Translation between European Languages. In P. Koehn, C. Monz, editors, *Proceedings on the Workshop on Statistical Machine Translation, WMT@HLT-NAACL 2006, New York City, NY, USA, June 8-9, 2006*, pp. 102–121. Association for Computational Linguistics, 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*, pp. 127–133, 2003.

[Kudo 18] T. Kudo: Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[Lakew & Lotito$^+$ 17a] 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 Conference on Spoken Language Translation*, pp. 113–119, Tokyo, Japan, Dec. 14-15 2017. International Workshop on Spoken Language Translation.

[Lakew & Lotito$^+$ 17b] S.M. Lakew, Q.F. Lotito, M. Turchi, M. Negri, M. Federico: FBK's Multilingual Neural Machine Translation System for IWSLT 2017. In S. Sakti, M. Utiyama, editors, *Proceedings of the 14th International Conference on Spoken Language Translation, IWSLT 2017, Tokyo, Japan, December 14-15, 2017*, pp. 35–41. International Workshop on Spoken Language Translation, 2017.

[Lample & Conneau$^+$ 18] G. Lample, A. Conneau, L. Denoyer, M. Ranzato: Unsupervised Machine Translation Using Monolingual Corpora Only. In *6th International Conference on Learn-*

*ing Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[Lan & Chen$^+$ 20] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[Lewis & Liu$^+$ 20] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, July 2020. Association for Computational Linguistics.

[Li & Li$^+$ 19] X. Li, G. Li, L. Liu, M. Meng, S. Shi: On the Word Alignment from Neural Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1293–1303, Florence, Italy, July 2019. Association for Computational Linguistics.

[Liu & Gu$^+$ 20] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, L. Zettlemoyer: Multilingual Denoising Pre-training for Neural Machine Translation. *Transactions of the Association for Computational Linguistics*, Vol. 8, pp. 726–742, 2020.

[Liu & Ott$^+$ 19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov: RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, Vol. abs/1907.11692, 2019.

[Luong & Pham$^+$ 15] T. Luong, H. Pham, C.D. Manning: Effective Approaches to Attention-based Neural Machine Translation. In L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, Y. Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 1412–1421. The Association for Computational Linguistics, 2015.

[Mathur & Baldwin$^+$ 20] N. Mathur, T. Baldwin, T. Cohn: Tangled up in BLEU: Reevaluating the Evaluation of Automatic Machine Translation Evaluation Metrics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4984–4997, Online, July 2020. Association for Computational Linguistics.

[Mi & Sankaran$^+$ 16] H. Mi, B. Sankaran, Z. Wang, A. Ittycheriah: Coverage Embedding Models for Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 955–960, Austin, Texas, Nov. 2016. Association for Computational Linguistics.

[Michel & Levy$^+$ 19] P. Michel, O. Levy, G. Neubig: Are Sixteen Heads Really Better than One? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett, editors, *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc., 2019.

[Michel & Schlüter$^+$ 20] W. Michel, R. Schlüter, H. Ney: Early Stage LM Integration Using Local and Global Log-Linear Combination. In *Interspeech*, pp. 3605–3609, Shanghai, China, Oct. 2020.

[Mikolov & Chen$^+$ 13] T. Mikolov, K. Chen, G. Corrado, J. Dean: Efficient Estimation of Word Representations in Vector Space. In Y. Bengio, Y. LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, May 2013.

[Murray & Chiang 18] K. Murray, D. Chiang: Correcting Length Bias in Neural Machine Translation. In O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, C. Monz, M. Negri, A. Névéol, M.L. Neves, M. Post, L. Specia, M. Turchi, K. Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pp. 212–223. Association for Computational Linguistics, 2018.

[Ney 01] H. Ney: Stochastic Modelling: From Pattern Classification to Language Translation. In *Proceedings of the ACL 2001 Workshop on Data-Driven Methods in Machine Translation*, 2001.

[Nguyen & Salazar 19] T.Q. Nguyen, J. Salazar: Transformers without Tears: Improving the Normalization of Self-Attention. In J. Niehues, R. Cattoni, S. Stüker, M. Negri, M. Turchi, T. Ha, E. Salesky, R. Sanabria, L. Barrault, L. Specia, M. Federico, editors, *Proceedings of the 16th International Conference on Spoken Language Translation, IWSLT 2019, Hong Kong, November 2-3, 2019*. Association for Computational Linguistics, 2019.

[Nix 19] A. Nix: Multitask Learning for Neural Machine Translation. Master's thesis, RWTH Aachen University, Department of Computer Science, 2019.

[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, 03 2003.

[Ott & Edunov$^+$ 18] M. Ott, S. Edunov, D. Grangier, M. Auli: Scaling Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation, Volume 1: Research Papers*, pp. 1–9, Belgium, Brussels, October 2018. Association for Computational Linguistics.

[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*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[Parcheta & Sanchis-Trilles$^+$ 19] Z. Parcheta, G. Sanchis-Trilles, F. Casacuberta: Filtering of Noisy Parallel Corpora Based on Hypothesis Generation. In O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, A. Martins, C. Monz, M. Negri, A. Névéol, M.L. Neves, M. Post, M. Turchi, K. Verspoor, editors, *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 3: Shared Task Papers, Day 2*, pp. 282–288. Association for Computational Linguistics, 2019.

[Parmar & Vaswani$^+$ 18] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, D. Tran: Image Transformer. In J. Dy, A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80 of *Proceedings of Machine Learning Research*, pp. 4055–4064. PMLR, 10–15 Jul 2018.

[Pennington & Socher$^+$ 14] J. Pennington, R. Socher, C.D. Manning: Glove: Global Vectors for Word Representation. In A. Moschitti, B. Pang, W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1532–1543. ACL, October 2014.

[Peter 20] J.T. Peter: *Exploration of Alignment Concepts to Bridge the Gap between Phrase-based and Neural Machine Translation.* Ph.D. thesis, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, July 2020.

[Peter & Guta$^+$ 17] J.T. Peter, A. Guta, T. Alkhouli, P. Bahar, J. Rosendahl, N. Rossenbach, M. Graça, N. Hermann: The RWTH Aachen University English-German and German-English Machine Translation System for WMT 2017. In *EMNLP 2017 Second Conference on Machine Translation*, Copenhagen, Denmark, Sept. 2017.

[Peters & Neumann$^+$ 18] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer: Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Peters & Ruder$^+$ 19] M.E. Peters, S. Ruder, N.A. Smith: To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. In I. Augenstein, S. Gella, S. Ruder, K. Kann, B. Can, J. Welbl, A. Conneau, X. Ren, M. Rei, editors, *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019*, pp. 7–14. Association for Computational Linguistics, August 2019.

[Petrick 20] F. Petrick: Extensions of Encoder-Decoder Attention for Neural Machine Translation. Master's thesis, RWTH Aachen University, Department of Computer Science, 2020.

[Petrick & Rosendahl$^+$ 22] F. Petrick, J. Rosendahl, C. Herold, H. Ney: Locality-Sensitive Hashing for Long Context Neural Machine Translation. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pp. 32–42, Dublin, Ireland (in-person and online), May 2022. Association for Computational Linguistics.

[Pham & Sperber$^+$ 17] N. Pham, M. Sperber, E. Salesky, T. Ha, J. Niehues, A. Waibel: KIT's Multilingual Neural Machine Translation systems for IWSLT 2017. In S. Sakti, M. Utiyama, editors, *Proceedings of the 14th International Conference on Spoken Language Translation, IWSLT 2017, Tokyo, Japan, December 14-15, 2017*, pp. 42–47. International Workshop on Spoken Language Translation, 2017.

[Popel & Bojar 18] M. Popel, O. Bojar: Training Tips for the Transformer Model. *Prague Bull. Math. Linguistics*, Vol. 110, pp. 43–70, 2018.

[Popović 15] M. Popović: chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pp. 392–395, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics.

[Post 18] M. Post: A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.

[Pratt & Mostow$^+$ 91] L.Y. Pratt, J. Mostow, C.A. Kamm: Direct Transfer of Learned Information Among Neural Networks. In T.L. Dean, K.R. McKeown, editors, *Proceedings of the 9th National Conference on Artificial Intelligence, Anaheim, CA, USA, July 14-19, 1991, Volume 2*, pp. 584–589. AAAI Press / The MIT Press, 1991.

[Qi & Sachan$^+$ 18] Y. Qi, D.S. Sachan, M. Felix, S. Padmanabhan, G. Neubig: When and Why Are Pre-Trained Word Embeddings Useful for Neural Machine Translation? In M.A. Walker, H. Ji, A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pp. 529–535. Association for Computational Linguistics, 2018.

[Qiu & Ma$^+$ 20] J. Qiu, H. Ma, O. Levy, W.t. Yih, S. Wang, J. Tang: Blockwise Self-Attention for Long Document Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2555–2565, Online, Nov. 2020. Association for Computational Linguistics.

[Qiu & Sun$^+$ 20] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, X. Huang: Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, Vol. 63, No. 10, pp. 1872–1897, 2020.

[Raffel & Shazeer$^+$ 20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P.J. Liu: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, Vol. 21, pp. 140:1–140:67, 2020.

[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*, pp. 383–391, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[Rei & Stewart$^+$ 20] R. Rei, C. Stewart, A.C. Farinha, A. Lavie: COMET: A Neural Framework for MT Evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2685–2702, Online, Nov. 2020. Association for Computational Linguistics.

[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 *ACL 2019 Fourth Conference on Machine Translation*, pp. 349–355, Florence, Italy, July 2019.

[Rosendahl & Herold$^+$ 21] J. Rosendahl, C. Herold, F. Petrick, H. Ney: Recurrent Attention for the Transformer. In *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, pp. 62–66, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[Rosendahl & Tran$^+$ 19] J. Rosendahl, V.A.K. Tran, W. Wang, H. Ney: Analysis of Positional Encodings for Neural Machine Translation. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong, Nov. 2-3 2019. Association for Computational Linguistics.

[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 *EMNLP 2018 Third Conference on Machine Translation*, pp. 946–954, Brussels, Belgium, October 2018.

[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.

[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 *EMNLP 2018 Third Conference on Machine Translation*, pp. 496–503, Brussels, Belgium, Oct. 2018.

[Schwenk 08] H. Schwenk: Investigations on large-scale lightly-supervised training for statistical machine translation. In *Proceedings of the 5th International Workshop on Spoken Language Translation: Papers*, pp. 182–189, Waikiki, Hawaii, Oct. 20-21 2008.

[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: Volume 2, Shared Task Papers*, pp. 371–376, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[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, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 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 (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[Shaw & Uszkoreit$^+$ 18] P. Shaw, J. Uszkoreit, A. Vaswani: Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Shen & Cheng$^+$ 16] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, Y. Liu: Minimum Risk Training for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

[Snover & Dorr$^+$ 06] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul: A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pp. 223–231, 2006.

[Srivastava & Hinton$^+$ 14] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 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 O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, C. Monz, M. Negri, A. Névéol, M.L. Neves, M. Post, L. Specia, M. Turchi, K. Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pp. 204–211. Association for Computational Linguistics, 2018.

[Stahlberg & de Gispert$^+$ 18] F. Stahlberg, A. de Gispert, B. Byrne: The University of Cambridge's Machine Translation Systems for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pp. 504–512, Belgium, Brussels, Oct. 2018. Association for Computational Linguistics.

[Sun & Cao$^+$ 20] S. Sun, Z. Cao, H. Zhu, J. Zhao: A Survey of Optimization Methods From a Machine Learning Perspective. *IEEE Transactions on Cybernetics*, Vol. 50, No. 8, pp. 3668–3681, 2020.

[Sutskever & Vinyals$^+$ 14] I. Sutskever, O. Vinyals, Q.V. Le: Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger, editors, *Advances in Neural Information Processing Systems*, Vol. 27. Curran Associates, Inc., 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 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[Tang & Müller$^+$ 18] G. Tang, M. Müller, A. Rios, R. Sennrich: Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures. In E. Riloff, D. Chiang, J. Hockenmaier, J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 4263–4272. Association for Computational Linguistics, 2018.

[Tay & Dehghani$^+$ 22] Y. Tay, M. Dehghani, D. Bahri, D. Metzler: Efficient Transformers: A Survey, apr 2022. Just Accepted.

[Taylor 53] W.L. Taylor: "Cloze procedure": A new tool for measuring readability. *Journalism quarterly*, Vol. 30, No. 4, pp. 415–433, 1953.

[Tiedemann 12] J. Tiedemann: Parallel Data, Tools and Interfaces in OPUS. In N.C.C. Chair), K. Choukri, T. Declerck, M.U. Dogan, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[Tokarchuk & Rosendahl$^+$ 21] E. Tokarchuk, J. Rosendahl, W. Wang, P. Petrushkov, T. Lancewicki, S. Khadivi, H. Ney: Integrated Training for Sequence-to-Sequence Models Using Non-Autoregressive Transformer. In *International Workshop on Spoken Language Translation*, online, Aug. 2021.

[Tu & Lu$^+$ 16] Z. Tu, Z. Lu, Y. Liu, X. Liu, H. Li: Modeling Coverage for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–85, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[Vaswani & Shazeer$^+$ 17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin: Attention is All you Need. In I. Guyon, U. von Luxburg, S. Bengio, H.M. Wallach, R. Fergus, S.V.N. Vishwanathan, R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

[Vilar & Freitag$^+$ 23] D. Vilar, M. Freitag, C. Cherry, J. Luo, V. Ratnakar, G.F. Foster: Prompting PaLM for Translation: Assessing Strategies and Performance. In A. Rogers, J.L. Boyd-Graber, N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 15406–15427. Association for Computational Linguistics, 2023.

[Vogel & Ney$^+$ 96] S. Vogel, H. Ney, C. Tillmann: HMM-Based Word Alignment in Statistical Translation. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*, August 1996.

[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*, pp. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics.

[Wang 23] W. Wang: *Neural Hidden Markov Model for Machine Translation.* Ph.D. thesis, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, 2023.

[Wang & Cho$^+$ 20] C. Wang, K. Cho, J. Gu: Neural Machine Translation with Byte-Level Subwords. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 9154–9160. AAAI Press, 2020.

[Wang & Gong$^+$ 18] M. Wang, L. Gong, W. Zhu, J. Xie, C. Bian: Tencent Neural Machine Translation Systems for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pp. 522–527, Belgium, Brussels, Oct. 2018. Association for Computational Linguistics.

[Wang & Li$^+$ 18] Q. Wang, B. Li, J. Liu, B. Jiang, Z. Zhang, Y. Li, Y. Lin, T. Xiao, J. Zhu: The NiuTrans Machine Translation System for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pp. 528–534, Belgium, Brussels, Oct. 2018. Association for Computational Linguistics.

[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. 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. *CoRR*, Vol. abs/1609.08144, 2016.

[Wuebker & Huck$^+$ 12] J. Wuebker, M. Huck, S. Peitz, M. Nuhn, M. Freitag, J. Peter, S. Mansour, H. Ney: Jane 2: Open Source Phrase-based and Hierarchical Statistical Machine Translation. In M. Kay, C. Boitet, editors, *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Demonstration Papers, 8-15 December 2012, Mumbai, India*, pp. 483–492. Indian Institute of Technology Bombay, December 2012.

[Wynands 21] N.P. Wynands: Global Normalization for Statistical Machine Translation. Master's thesis, RWTH Aachen University, Department of Computer Science, September 2021.

[Wynands & Michel$^+$ 22] N. Wynands, W. Michel, J. Rosendahl, R. Schlüter, H. Ney: Efficient Sequence Training of Attention Models Using Approximative Recombination. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore, 23-27 May 2022*, pp. 8002–8006. IEEE, 2022.

[Xiong & Yang$^+$ 20] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, T. Liu: On Layer Normalization in the Transformer Architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Vol. 119 of *Proceedings of Machine Learning Research*, pp. 10524–10533. PMLR, July 2020.

[Yang & Dai$^+$ 19] Z. Yang, Z. Dai, Y. Yang, J.G. Carbonell, R. Salakhutdinov, Q.V. Le: XLNet: Generalized Autoregressive Pretraining for Language Understanding. In H.M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E.B. Fox, R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 5754–5764, December 2019.

[You & Sun$^+$ 20] W. You, S. Sun, M. Iyyer: Hard-Coded Gaussian Attention for Neural Machine Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7689–7700, Online, July 2020. Association for Computational Linguistics.

[Zenkel & Wuebker$^+$ 20] T. Zenkel, J. Wuebker, J. DeNero: End-to-End Neural Word Alignment Outperforms GIZA++. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1605–1617, Online, July 2020. Association for Computational Linguistics.

[Zens & Och$^+$ 02] R. Zens, F.J. Och, H. Ney: Phrase-Based Statistical Machine Translation. In M. Jarke, J. Koehler, G. Lakemeyer, editors, *KI 2002: Advances in Artificial Intelligence, 25th Annual German Conference on AI, KI 2002, Aachen, Germany, September 16-20, 2002, Proceedings*, Vol. 2479 of *Lecture Notes in Computer Science*, pp. 18–32. Springer, September 2002.

[Zeyer & Bahar$^+$ 19] A. Zeyer, P. Bahar, K. Irie, R. Schlüter, H. Ney: A Comparison of Transformer and LSTM Encoder Decoder Models for ASR. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019, Singapore, December 14-18, 2019*, pp. 8–15. IEEE, December 2019.

[Zhang & Han$^+$ 19] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, Q. Liu: ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1441–1451, Florence, Italy, July 2019. Association for Computational Linguistics.

[Zhang & Kishore$^+$ 20] T. Zhang, V. Kishore, F. Wu, K.Q. Weinberger, Y. Artzi: BERTScore: Evaluating Text Generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[Zhang & Yu$^+$ 20] Y. Zhang, X. Yu, Z. Cui, S. Wu, Z. Wen, L. Wang: Every Document Owns Its Structure: Inductive Text Classification via Graph Neural Networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 334–339, Online, July 2020. Association for Computational Linguistics.

[Zhao & Ni$^+$ 20] Y. Zhao, C. Ni, C. Leung, S.R. Joty, E.S. Chng, B. Ma: Cross Attention with Monotonic Alignment for Speech Transformer. In H. Meng, B. Xu, T.F. Zheng, editors, *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pp. 5031–5035. ISCA, October 2020.

[Zhou & Zhou$^+$ 21] S. Zhou, T. Zhou, B. Wei, Y. Luo, Y. Mu, Z. Zhou, C. Wang, X. Zhou, C. Lv, Y. Jing, L. Wang, J. Zhang, C. Huang, Z. Yan, C. Hu, B. Li, T. Xiao, J. Zhu: The NiuTrans Machine Translation Systems for WMT21. In *Proceedings of the Sixth Conference on Machine Translation*, pp. 265–272, Online, November 2021. Association for Computational Linguistics.

[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*, pp. 1568–1575, Austin, Texas, Nov. 2016. Association for Computational Linguistics.