Moving Beyond Parallel Data for Neural Machine Translation

Anna Currey

Doctor of Philosophy
Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh
2019

Abstract

The goal of neural machine translation (NMT) is to build an end-to-end system that automatically translates sentences from the source language to the target language. Neural machine translation has become the dominant paradigm in machine translation in recent years, showing strong improvements over prior statistical methods in many scenarios. However, neural machine translation relies heavily on parallel corpora for training; even for two languages with abundant monolingual resources (or with a large number of speakers), such parallel corpora may be scarce. Thus, it is important to develop methods for leveraging additional types of data in NMT training. This thesis explores ways of augmenting the parallel training data of neural machine translation with non-parallel sources of data. We concentrate on two main types of additional data: monolingual corpora and structural annotations. First, we propose a method for adding target-language monolingual data into neural machine translation in which the monolingual data is converted to parallel data through copying. Thus, the NMT system is trained on two tasks: translation from source language to target language, and autoencoding the target language. We show that this model achieves improvements in BLEU score for low- and medium-resource setups. Second, we consider the task of zero-resource NMT, where no source $\leftrightarrow$ target parallel training data is available, but parallel data with a pivot language is abundant. We improve these models by adding a monolingual corpus in the pivot language, translating this corpus into both the source and the target language to create a pseudo-parallel source $\leftrightarrow$ target corpus. In the second half of this thesis, we turn our attention to syntax, introducing methods for adding syntactic annotation of the source language into neural machine translation. In particular, our multi-source model, which leverages an additional encoder to inject syntax into the NMT model, results in strong improvements over non-syntactic NMT for a high-resource translation case, while remaining robust to unparsed inputs. We also introduce a multi-task model that augments the transformer architecture with syntax; this model improves translation across several language pairs. Finally, we consider the case where no syntactic annotations are available (such as when translating from very low-resource languages). We introduce an unsupervised hierarchical encoder that induces a tree structure over the source sentences based solely on the downstream task of translation. Although the resulting hierarchies do not resemble traditional syntax, the model shows large improvements in BLEU for low-resource NMT.

Lay Summary

Machine translation, the process of automatically translating text from a source language (e.g. English) to a target language (e.g. French), has achieved impressive results in recent years. However, modern machine translation methods rely heavily on parallel data – millions of sentences translated from the source to the target language. Such parallel data is not readily available for most pairs of source and target languages. The goal of this thesis is to explore ways of using other types of data to improve the translations generated by machine translation systems. We consider two main types of data. The first is text in a single language, which for many languages is readily available. This text could include news articles, government proceedings, books, online posts, etc. We show that teaching a system to copy and translate text simultaneously can help it improve translation when not much parallel data is available. We also introduce a method where translation from the source language into the target language can be improved using text from a third pivot language by first automatically translating the sentences in the pivot language into both the source and the target language and then using the resulting (pivot, source, target) sentence triples as additional training data. The second type of data we explore in this thesis is information about the sentence structure, such as the part-of-speech for each word and the phrases of the sentence. We train our machine translation systems with this additional information about the structure of the sentences in the source language, which results in higher-quality translations in the target language. Additionally, we propose a model where the machine translation system automatically infers information about the sentence structure while also learning to translate. Both types of data – additional text in one language only, and information about the linguistic structure of the languages – help to improve machine-generated translations in cases where little parallel training data is available.

Acknowledgements

The work in chapter 3 was conducted within the scope of the Horizon 2020 Innovation Action *Health in My Language*, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644402.

GPU time was supported by Microsoft's donation of Azure credits to The Alan Turing Institute. This work was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

The research in chapters 3 and 5 was also funded by the Amazon Academic Research Awards program.

The research in chapter 7 is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract # FA8650-17-C-9117. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

I would also like to thank my principal supervisor (Kenneth Heafield), my thesis examiners (Alexandra Birch and Marine Carpuat), my second supervisor (Steve Renals), and my colleagues in the ILCC and in the StatMT group. On a personal note, I would like to thank my family and friends for their love and support.

Declaration

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Machine translation (MT) is the process of automatically translating a piece of text from one language, the source language, into another language, the target language. The current dominant paradigm for machine translation is neural machine translation (NMT), which uses two neural networks: one (the encoder) reads in a sentence in the source language, and the other (the decoder) generates its translation in the target language (Cho et al., 2014b; Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014).

Modern machine translation methods are most often trained using a parallel corpus, which is a corpus consisting of text in the source language and their translations in the target language (usually aligned on the sentence level). Such corpora can be created from parliamentary proceedings of multilingual governments (e.g. Koehn, 2005) or from multilingual news services (e.g. Tyers and Alperen, 2010), for example. Although large parallel corpora exist, they are often in specific domains (e.g. government or news) and only between a few languages (e.g. bilingual corpora with English, or languages of the European Union). Less common domains or language pairs may not have much parallel data available, making it hard to train strong NMT systems.

Neural machine translation has achieved impressive quality on very high-resource tasks where large parallel corpora are available (Hassan et al., 2018; Wu et al., 2016). However, its performance on lower-resource tasks tends to be far worse than on high-resource tasks and can even lag behind older statistical machine translation (SMT) methods when there is a domain mismatch (Koehn and Knowles, 2017). Since NMT typically relies on parallel corpora, which are harder to come by than monolingual corpora, it is not always possible to simply acquire more training data.

Although parallel data may be relatively scarce for a given language pair or domain, several other sources of data are available. There are many possible types of non-parallel data, including:

- Monolingual data (in the source language, in the target language, or in a third language e.g. a related language)
- Parallel corpora between the source/target language and another language (many language pairs have little direct parallel data between them, but abundant source $\leftrightarrow$ English and target $\leftrightarrow$ English parallel data)
- Linguistic annotations (morphological, syntactic, semantic, etc.)

However, neural machine translation is usually trained to maximize the probability of a sentence in the target language, given its corresponding sentence in the source language. Thus, the standard NMT training objective requires a (source, target) sentence pair, and it is not immediately obvious how to train on monolingual sentences or text in other languages. In addition, the encoder and decoder of standard neural machine translation are usually long short-term memory networks (LSTMs), convolutional neural networks (CNNs), or transformer networks, all of which are constructed to take a sequence of words as input. Using information about linguistic structure could potentially aid neural machine translation, but doing so would require changes to these standard architectures or to the linguistic annotations themselves. The goal of this thesis is thus to explore the question of how best to incorporate these existing non-parallel data types into neural machine translation to improve its performance, particularly in low-resource scenarios.

For the purposes of this thesis, we draw a distinction between a low-resource language (a language for which few monolingual resources, including corpora and linguistic annotators, are available) and a low-resource language pair (a language pair that has a small amount of parallel data or no parallel data at all). A low-resource language pair does not necessarily consist of two low-resource languages; it may be made up of a high- and a low-resource language (e.g. English and Tagalog) or even two high-resource languages (e.g. German and Russian), as long as the two languages have little parallel data. In this thesis, we focus mainly on low-resource language pairs in which at least one of the languages is a high-resource language (although we consider other scenarios as well).

1.2 Main Hypotheses

We explore two main types of non-parallel data in this thesis. The first is **monolingual corpora**, which are particularly important because of their ubiquity (for many languages) compared to other potential resources. We expect monolingual data to be helpful partly because the NMT encoder and decoder are modified language models (Ramachandran et al., 2017), and language models are usually trained on monolingual data. We propose ways of incorporating monolingual data from the source and target language, as well as from a third high-resource pivot language, into NMT training. In this thesis, we explore two main hypotheses relating to monolingual data:

- Adding target-side monolingual data using an autoencoding objective to NMT, in which the system simultaneously learns to translate and copy, can lead to improvements in low-resource NMT without requiring a costly pre-training step. This hypothesis is evaluated in chapter 3.

- In zero-resource neural machine translation, where no direct parallel training data is available, monolingual data in the pivot language can be leveraged to improve translation performance. This hypothesis is evaluated in chapter 4.

The second type of non-parallel data that we use in this thesis is **syntactic information about the source language**. Syntactic parsers exist and achieve reasonably good accuracy for most high-resource languages. Thus, when translating out of a high-resource source language, it is relatively easy to generate relevant syntactic annotations for the source side of the parallel corpus by applying an automatic parser to the source data. We consider source syntax because neural machine translation tends to improve over SMT much more in fluency than in adequacy (Castilho et al., 2017; Koehn and Knowles, 2017), and we speculate that a better understanding of the source language is important for improving translation adequacy. Our main hypotheses regarding syntactic annotations are:

- Linearized versions of consistuency parses can be used within existing NMT architectures to improve NMT performance without requiring data to be parsed at inference time. This hypothesis is evaluated in chapters 5 and 6.

- Inducing an unsupervised binary tree over the source sentence is an effective way of adding structure for very low-resource source languages where no parser is available. This hypothesis is evaluated in chapter 7.

1.3 Thesis Structure

The remainder of this thesis will be structured as follows.

- **Chapter 2** gives an overview of the relevant literature, starting with an explanation of neural machine translation. We also review related work on using monolingual data in NMT, improving NMT with multilingual data, and injecting syntactic structure into NMT.

- **Chapter 3** is the first of two chapters that address augmenting the parallel NMT training corpus with monolingual data. In this chapter, we introduce a *copied corpus* method for adding target-side monolingual data into standard NMT training that is able to leverage monolingual data without any pre-training. This method starts with a monolingual target-language corpus, which we convert into a bitext by copying it, making the source and target sides identical. This pseudo-parallel corpus is then mixed with the true parallel data and the back-translated monolingual data; the resulting mixture is used to train the NMT system. We show that this method results in performance improvements for low- and medium-resource language pairs, partly due to improved accuracy at copying words that should be identical in the source and target sentences. This chapter also contains further analysis of the copied corpus method and the conditions under which it is beneficial.

- **Chapter 4** also addresses monolingual data for NMT; this time, the monolingual data is in a pivot language. We consider a zero-resource neural machine translation framework in which we are given a source $\leftrightarrow$ pivot parallel corpus and a pivot $\leftrightarrow$ target parallel corpus, but no direct source $\leftrightarrow$ target parallel training data. Our insight is that we can use monolingual data in the pivot language to improve NMT performance; this is important because the pivot language is often the highest-resource language of the three. We show that the monolingual pivot data can be used to generate a pseudo-parallel source $\leftrightarrow$ target corpus, which can then be used to fine-tune the NMT model or re-train it from scratch. The resulting systems yield strong improvements in direct translation over both direct and pivot-based baselines.

- **Chapter 5** considers the use of syntactic annotations as an augmentation to the NMT parallel training data. We parse the source side of the parallel corpus using an off-the-shelf parser and introduce a method for training NMT on these parses that consists of using two encoders simultaneously, one for parsed sentences and the other for unparsed sentences. This method is advantageous over previous work on syntactic NMT because it would be easy to apply to new architectures, it does not fail when no parsed input is available, and it performs reasonably well on long sentences. We also show preliminary work on an extension to this multi-source model, the shared encoder model, which trains a single encoder on both parsed and unparsed source sentences. The shared encoder model yields even further improvements over the baseline.

- **Chapter 6** builds on chapter 5 by adding source-side parses into transformer-based NMT. We study a multi-task system that learns to both translate and parse the source sentences with a single encoder and decoder. We perform cross-lingual analysis for translation from English into several target languages, finding that the multi-task system consistently improves over a non-syntactic baseline. On a high-resource task, however, the non-syntactic model outperforms the multi-task syntactic model. We further analyze the cross-lingual results, finding that target language family does not seem to have an effect on translation performance but may have an effect on parsing performance.

- **Chapter 7** extends our work on syntactic NMT to very low-resource scenarios. For low-resource source languages, there is often no available parser, so the models proposed in chapters 5 and 6 cannot be applied. Thus, we introduce an unsupervised hierarchical encoder based on a Gumbel tree-LSTM (Choi et al., 2018) that induces a tree structure over the source sentences without any syntactic supervision. This encoder results in strongly improved performance compared to standard NMT for low- and very low-resource language pairs, even though it does not seem to learn linguistic information.

- **Chapter 8** is our conclusion chapter. We summarize the thesis and give some ideas for possible future work.

1.4 Contributions

This thesis makes the following contributions:

- A method for adding a monolingual target corpus to NMT that improves translation performance for low- and medium-resource language pairs, partially due to improved accuracy on words that are identical in the source and target sentences. This work is based on Currey et al. (2017).

- A novel source of data, pivot-language monolingual data, for zero-resource neural machine translation, as well as several methods for leveraging this data to improve direct zero-resource translation performance.

- A way of adding source-side syntactic parses into NMT that is applicable to any architecture, does not require parsed input sentences at inference time, and improves translation on long sentences. This work is based on Currey and Heafield (2018a).

- Two methods for adding syntactic information to transformer-based NMT using linearized constituency parses of the source sentence, as well as experiments showing that syntactic annotations can improve low-resource translation for 21 diverse target languages. This work is baesd on Currey and Heafield (2019).

- Experiments showing that inducing an unsupervised hierarchical structure over the source sentences can improve LSTM-based neural machine translation in very low-resource scenarios without requiring any external syntactic parser. This work is based on Currey and Heafield (2018b).

Chapter 2

Background

2.1 Introduction

We start this chapter with a brief overview of neural machine translation in order to facilitate understanding of the rest of this thesis. The remainder of this chapter discusses various strategies for incorporating additional non-parallel data into neural machine translation, focusing on monolingual data (section 2.3), multilingual corpora (section 2.4), and syntactic annotations (section 2.5).

2.2 Neural Machine Translation

We use neural machine translation as the basis for all of the experiments in this thesis. This section contains a brief introduction to how neural machine translation works, as well as an overview of some of its limitations.

2.2.1 Sequence-to-Sequence Models

Neural machine translation is a neural network-based machine translation paradigm that uses a sequence-to-sequence (seq2seq) model as its basis (Cho et al., 2014b; Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014). Seq2seq models consist of two neural networks joined together; they are trained to match an input sequence to an output sequence. In NMT, the first neural network is the *encoder*; it reads in the sequence of source words and encodes it as a vector. The second neural network is the *decoder*; it takes as input the source encoding and predicts the probability of the target sentence, conditioned on the source sentence. Figure 2.1 depicts the basic

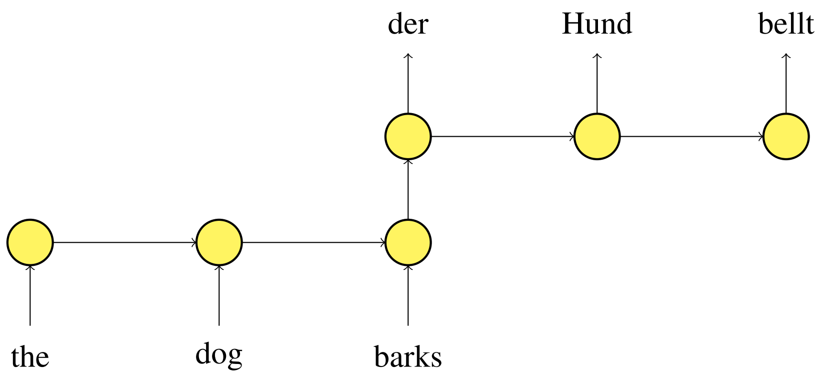


Figure 2.1: Basic sequence-to-sequence model for neural machine translation. The bottom half of the figure corresponds to the encoder, and the top half corresponds to the decoder.

sequence-to-sequence architecture for NMT.

2.2.2 Training Objective

Neural machine translation is often trained on corpora of parallel sentences; it attempts to directly model the probability $P(t|s)$ of a target sentence t given a source sentence s. Thus, the standard training objective in NMT, given a parallel corpus (S,T) of N training sentences (s_i,t_i), is to minimize the negative log-likelihood of the training corpus:

$$L((S,T),\theta) = -\frac{1}{N}\sum_{i=1}^{N}\log P(t_i|s_i,\theta) \tag{2.1}$$

where θ is the model parameters. Since standard NMT optimization is done on the basis of sentence pairs, parallel data is usually important for NMT training.

2.2.3 Attention Mechanism

The basic seq2seq models described in section 2.2.1 achieve good performance when translating short sentences. However, their performance deteriorates significantly as sentence length increases (Bahdanau et al., 2015; Cho et al., 2014a) because the encoder maps the source sentence into a single vector of a fixed length, which can be insufficient for modeling long sentences.

Bahdanau et al. (2015) propose adding an attention mechanism to neural machine translation in order to remedy this issue. This is done by allowing the decoder to look at different parts of the source sentence at each timestep, rather than limiting the source

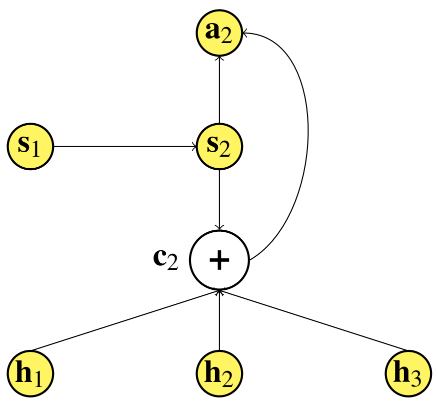


Figure 2.2: Illustration of the attention calculation at timestep $j = 2$ in the decoder. The context vector $\mathbf{c}_j$ is calculated based on the current decoder hidden state $\mathbf{s}_j$ and all the encoder hidden states $\mathbf{h}_i$. The final attention vector $\mathbf{a}_j$ is then calculated from the decoder hidden state and the context vector.

sentence representation to a single fixed-length vector. Figure 2.2 gives an overview of the attention process at a given timestep in the decoder.

In attention-based NMT, the encoder first embeds the source sentence to create a single vector representation of the sentence, as before. However, this vector is only used to initialize the decoder. At each timestep j in the decoder, a context vector $\mathbf{c}_j$ is taken as additional input. This context vector is calculated as a weighted sum over the hidden states $\mathbf{h}_i$ of the encoder:

$$\alpha_j(i) = \frac{exp(S(\mathbf{h}_i, \mathbf{s}_j))}{\sum_k exp(S(\mathbf{h}_k, \mathbf{s}_j))} \tag{2.2}$$

$$\mathbf{c}_j = \sum_i \alpha_j(i)\mathbf{h}_i \tag{2.3}$$

where $\mathbf{s}_j$ is the hidden state of the decoder and S is a scoring function. In the original formulation of attention by Bahdanau et al. (2015), the score is calculated using a feed-forward network over the encoder and decoder hidden states:

$$S(\mathbf{h}_i, \mathbf{s}_j) = \mathbf{v}^\top \tanh(\mathbf{W}_1\mathbf{h}_i + \mathbf{W}_2\mathbf{s}_j) \tag{2.4}$$

where $\mathbf{v}$, $\mathbf{W}_1$, and $\mathbf{W}_2$ are the parameters of the feed-forward neural network. Other scoring functions, such as a dot product, are also possible (Luong et al., 2015a). Finally, the attention vector $\mathbf{a}_j$ is the concatenation of the context vector and the current decoder hidden state:

$$\mathbf{a}_j = \tanh(\mathbf{W}_c[\mathbf{c}_j; \mathbf{s}_j]) \tag{2.5}$$

where $\mathbf{W}_c$ is a weight matrix. This attention vector can then be fed into the softmax function to predict the next word in the target sentence.

2.2.4 Model Architectures

Seq2seq models are relatively flexible in that the two neural networks they consist of can be constructed in different ways. Several different encoder and decoder architectures for NMT have been proposed, including convolutional neural networks (Gehring et al., 2017a,b), recurrent neural networks (Cho et al., 2014b), and fully attentional transformer networks (Vaswani et al., 2017). In this thesis, we run our experiments on recurrent neural network (chapters 3, 5, and 7) and transformer (chapters 4 and 6) architectures; as such, we discuss those two architectures in more detail here.

RNN-Based Neural Machine Translation

A recurrent neural network (RNN) is a type of neural network that is particularly well-designed for modeling sequences. At each timestep j, the RNN hidden state $\mathbf{h}_j$ is calculated as a function of both the current input $\mathbf{x}_j$ and the previous timestep's hidden state $\mathbf{h}_{j-1}$:

$$\mathbf{h}_j = f(\mathbf{x}_j, \mathbf{h}_{j-1}) \tag{2.6}$$

This allows the current state to take the history into account, so that information from earlier in the sequence is not lost.

In practice, RNN cells are often augmented with gating mechanisms using either long short-term memory units (LSTMs; Hochreiter and Schmidhuber, 1997) or gated recurrent units (GRUs; Cho et al., 2014b). These gating mechanisms mitigate the vanishing gradient issue that makes RNNs difficult to train through back-propagation. At each timestep j, an LSTM cell contains an input gate $\mathbf{i}_j$, an output gate $\mathbf{o}_j$, and a forget gate $\mathbf{f}_j$ which are used to calculate the cell state $\mathbf{c}_j$ and hidden state $\mathbf{h}_j$ as follows:

$$\mathbf{f}_j = \sigma(\mathbf{W}_f \mathbf{x}_j + \mathbf{U}_f \mathbf{h}_{j-1} + \mathbf{b}_f) \tag{2.7}$$

$$\mathbf{i}_j = \sigma(\mathbf{W}_i \mathbf{x}_j + \mathbf{U}_i \mathbf{h}_{j-1} + \mathbf{b}_i) \tag{2.8}$$

$$\mathbf{o}_j = \sigma(\mathbf{W}_o \mathbf{x}_j + \mathbf{U}_o \mathbf{h}_{j-1} + \mathbf{b}_o) \tag{2.9}$$

$$\mathbf{c}_j = \mathbf{f}_j \odot \mathbf{c}_{j-1} + \mathbf{i}_j \odot \sigma(\mathbf{W}_c \mathbf{x}_j + \mathbf{U}_c \mathbf{h}_{j-1} + \mathbf{b}_c) \tag{2.10}$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \sigma(\mathbf{c}_j) \tag{2.11}$$

where $\mathbf{x}_j$ is the input at timestep j and $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are the weights and biases. The GRU is calculated in a similar manner, although it does not have an output gate or a cell state.

LSTMs and GRUs are commonly used as both the encoder and the decoder in neural machine translation. The encoder is often a bidirectional LSTM (biLSTM) or GRU (biGRU) that encodes both forward and backward versions of the source sentence (Bahdanau et al., 2015). The forward model reads in the sentence left-to-right and the backward model reads it in right-to-left; the forward and backward hidden states for each word are then concatenated together to create the final representation for the word. Thus, the final representation for each word contains information about what comes before and after it in the sentence.

Transformer Model

The transformer model (Vaswani et al., 2017) is a self-attentional neural network model that has recently outperformed RNN-based neural machine translation models in several scenarios (Bojar et al., 2018). Transformer-based NMT uses the same encoder-decoder-attention structure described in sections 2.2.1 and 2.2.3, but the internal architecture of the encoder and decoder is novel.

The transformer encoder is made up of several (often six) identical layers. Each encoder layer consists of two sub-layers: a multi-headed attention layer and a fully connected feed-forward neural network. These layers are stacked on top of each other to generate the final encoding of the source sentence. The transformer decoder has a similar structure, but with an additional layer to attend to the source sentence (similar to the standard attentional layer described in section 2.2.3).

The main innovation of the transformer architecture is the self-attention layer in both the encoder and the decoder; this is in addition to the standard attention in which the decoder attends to the encoder. In the self-attention encoder layers, every word in the source sentence attends to every other word in that sentence, and the resulting attention vector is used as the representation of that word in that layer. In practice, transformers use multi-headed attention, which means that multiple attention mechanisms are run for each word. The attention in the decoder works similarly, but words in the sentence after the current word are masked (so that the model cannot use words that have not yet been generated to generate the current word).

2.2.5 NMT with Subword Units

A major challenge for neural machine translation is the translation of rare words. Word-based machine translation operates on a fixed vocabulary that is necessarily much smaller than the vocabulary of the languages involved. At test time, new words may be introduced, so it is important for an MT system to be robust to words that were not seen during training. This is especially true in low-resource cases, where the training data may not be large enough to contain even relatively common words, and where common words may not appear often enough in the training data for the model to be able to learn a good representation for them.

One effective solution to this issue is to train neural machine translation on the subword level, rather than on the word level. Throughout this thesis, we break words into subwords using byte pair encoding (BPE), as introduced by Sennrich et al. (2016d). In this framework, a subword vocabulary is learned from the training data based on frequency as follows:

1. Initialize the subword vocabulary with the vocabulary of characters in the data.
2. Find the most frequently co-occurring pair of subwords within the words in the data and merge them into a single subword; add this subword to the vocabulary.
3. Iterate until the desired vocabulary size is achieved.

In the experiments described in this thesis, unless otherwise noted, we create a common subword vocabulary for both the source and the target languages by concatenating source and target training data together before learning the vocabulary.

2.2.6 Limitations of NMT

Neural machine translation can achieve high translation quality in several tasks (Bojar et al., 2017, 2018); however, NMT is not perfect. One of the main challenges facing NMT is that it requires a large amount of training data in order to achieve good performance. Although NMT does well in high-resource scenarios, Koehn and Knowles (2017) found that SMT beats NMT in low- and moderate-resource cases (up to about

15 million words of parallel data). Thus, systems that work well in high-resource cases cannot necessarily be directly applied to low-resource cases. In addition, NMT does poorly when there is a domain mismatch between training and test data, whereas SMT is better able to generalize to an unseen domain (Koehn and Knowles, 2017). Since it is important to be able to perform machine translation in cases where limited in-domain parallel data is available, figuring out ways of adding monolingual or multilingual data to low-resource NMT training is an interesting avenue for research. In this thesis, we explore these areas in chapters 3 and 4.

Another issue with neural machine translation is that it is not able to perfectly model source and target grammar, which can lead to inadequate or disfluent translations. Even though NMT has made gains over statistical machine translation in this regard (Bentivogli et al., 2016), there is still room for further improvement. Bentivogli et al. (2016) performed a fine-grained error analysis of English→German NMT outputs. They found that NMT performed strongly overall, but made errors on negation, which requires a deep understanding of the source sentence. This caused the NMT system to output fluent but inadequate translations. In addition, Sennrich (2017) introduced a corpus of contrastive translation pairs for English→German in order to analyze NMT performance on different linguistic phenomena. NMT had high accuracy overall, but did make some errors on negation and on finding the correct verb particle for separable German verbs. The transformer architecture also seems to have some trouble learning syntax for low-resource language pairs. Raganato and Tiedemann (2018) found that transformer encoders trained on high-resource NMT could achieve high accuracy on several grammatical tasks, whereas low-resource transformers did not do as well on these tasks. Thus, there is some evidence that neural machine translation (particularly low-resource neural machine translation) stands to benefit from an improved representation of syntax. We address adding source-side syntax to NMT in chapters 5 and 6, and in chapter 7, we add a hierarchical structure to low-resource NMT with the goal of mimicking syntax.

2.3 Neural Machine Translation with Monolingual Data

For many language pairs, abundant monolingual data is available in both the source and the target language but little parallel data exists. A natural questions is thus whether such source and target monolingual data can be used to augment the parallel corpus when training machine translation systems, creating a semi-supervised framework in which both labeled (i.e. parallel) and unlabeled (i.e. monolingual) data is used. This section describes three common ways of doing this: leveraging monolingual language models (2.3.1), creating pseudo-parallel corpora (2.3.2), and training models to reconstruct translations (2.3.3). Finally, in section 2.3.4, we discuss systems that use these techniques to perform fully unsupervised machine translation. Chapter 3 of this thesis contains our proposal of an efficient method for using target-side monolingual data in neural machine translation.

2.3.1 Language Models

Early attempts at incorporating target-language monolingual data into neural machine translation mirrored prior statistical machine translation methods by leveraging a separately trained target language model. Gulcehre et al. (2015) propose two methods for integrating the language model into neural machine translation: shallow fusion and deep fusion. In shallow fusion, the pre-trained language model is used during inference to rescore NMT outputs, whereas in deep fusion, the language model and the NMT model are fine-tuned together. The deep fusion model in particular is successful at improving NMT with monolingual data; however, later back-translation methods (described in section 2.3.2) give better improvements with a simpler training paradigm.

Similarly, Ramachandran et al. (2017) pre-train both source and target RNN language models and use the resulting models to initialize the NMT encoder and decoder, respectively. The resulting model achieves similar results to back-translation (Sennrich et al., 2016c). One issue with the language modeling methods described here is that they require a pre-training step to train the language model, thus potentially increasing overall training time. In chapter 3, we propose a lightweight method for adding monolingual data into NMT that does not require any pre-training.

2.3.2 Back-Translation

In this section, we discuss back-translation-based methods for adding monolingual data into NMT. Our copied corpus system (chapter 3) uses back-translation as a baseline and as an additional component to the model. In chapter 4, we propose using back-translation to incorporate pivot-language monolingual data into NMT.

Back-translation is one of the most successful methods for semi-supervised neural machine translation. Introduced by Sennrich et al. (2016c) based on similar techniques for SMT (Bertoldi and Federico, 2009; Bojar and Tamchyna, 2011), back-translation is a versatile and effective way of injecting target monolingual data into neural machine translation. In back-translation, a target $\rightarrow$ source NMT model is first trained. This trained model is then used to translate the target monolingual data into the source language. Finally, the resulting source' $\rightarrow$ target pseudo-parallel corpus (where the prime indicates machine-generated data) is combined with the parallel training data to train a source $\rightarrow$ target NMT model. Back-translation shows large gains over models trained on only parallel data and has the advantage that it can be applied to any NMT architecture. The main drawback of back-translation is the fact that it requires extra training time (to train the reverse NMT system and to back-translate the monolingual data); in chapter 3, we address this issue by proposing our copied corpus model, which requires no pre-training and which does almost as well as back-translation on low-resource NMT.

Several researchers have proposed extensions to back-translation. Niu et al. (2018) leverage a bidirectional (source $\leftrightarrow$ target) neural machine translation system to incorporate both source and target monolingual data into NMT. First, they train a bidirectional NMT system on only the parallel data. The bidirectional system is then used to back-translate the source and target monolingual data to create source' $\rightarrow$ target and target' $\rightarrow$ source pseudo-parallel data, which can then be used to improve the bidirectional system. In chapter 4, we leverage a similar model: we use a multi-directional NMT system to create source' $\leftrightarrow$ target' pseudo-parallel corpora from a pivot-language monolingual corpus.

Hoang et al. (2018) propose an iterative back-translation model, where separate source $\rightarrow$ target and target $\rightarrow$ source NMT systems are iteratively improved using back-translation and then used to generate additional back-translated data. Inspired by this work, we try iterating our zero-resource systems described in chapter 4, but we do not see strong improvements from running multiple iterations (this is described in more detail in section 4.4.2).

2.3.3 Dual Learning for Neural Machine Translation

Dual learning methods for NMT bear some relation to the copied corpus method we propose in chapter 3 in that both use an autoencoding task to incorporate monolingual data into NMT. The insight behind dual learning methods is that source $\rightarrow$ target and target $\rightarrow$ source machine translation systems are complementary and can form a closed loop, where translating a sentence from the source language to the target language and back to the source language should ideally result in the original source sentence. Dual learning methods use this intuition to inject source- and target-language monolingual data into NMT by adding a reconstruction-based training objective.

Cheng et al. (2016) concatenate source $\rightarrow$ target and target $\rightarrow$ source neural machine translation models to create a source $\rightarrow$ target $\rightarrow$ source autoencoder (and similarly for target $\rightarrow$ source $\rightarrow$ target). In addition to the standard NMT objective trained on parallel data, they add a reconstruction objective trained on monolingual source and target data: for a given sentence from the monolingual corpus, they maximize the likelihood of getting the same sentence after it is fed through both NMT systems. Niu et al. (2019) improve over this method by using the straight-through Gumbel softmax estimator (Jang et al., 2017) to sample the intermediate translations in a differentiable way, so that training can be done end-to-end. They also use a single bidirectional neural machine translation system for their autoencoder, rather than two separate systems; however, they see only relatively small improvements from this method (compared to a baseline without monolingual data). The dual learning technique proposed by He et al. (2016) is similar to the method of Cheng et al. (2016), but it uses reinforcement learning to train the model to maximize fluency of the translations and reconstruction of the monolingual data. Our copied corpus system is also trained to reconstruct monolingual sentences, but without the two-step process of Cheng et al. (2016) and without requiring reinforcement learning. This makes it easier to train, because only one NMT system is required; however, we find in section 3.4.2 that the copied corpus method is not an effective strategy for using source-side monolingual data, unlike the NMT models trained with a reconstruction objective (although the copied corpus method does yield improvements from using target-language monolingual data in low-resource NMT).

2.3.4 Machine Translation Without Parallel Data

In chapter 3 of this thesis, we propose ways of using target and source monolingual data to improve neural machine translation when little parallel data is available. However, recent unsupervised machine translation methods (Artetxe et al., 2018; Lample et al., 2018) have shown that it is possible to train a neural machine translation system using only monolingual source and target data, with no parallel data at all.

These systems work by first training word embeddings for each language on the monolingual data, then aligning the two embedding spaces using adversarial training (Conneau et al., 2018). These aligned word embeddings are then used as a bilingual dictionary to translate the sentences in one language word-by-word into the other language, resulting in an initial pseudo-parallel corpus. Then, an encoder-decoder model is trained using two objectives. The first is denoising autoencoding: given a corrupt version of a sentence from the monolingual data, the model tries to reconstruct the true version. This is similar to our copied corpus method in chapter 3, where we also train with an autoencoding objective, although we do not use noisy source data. The second objective is machine translation on the pseudo-parallel corpus. Additionally, a discriminator that tries to detect the original language of a sentence based on its encoding is used to encourage the model to encode sentences in the same embedding space regardless of language. Finally, training is done iteratively: the currently trained model is used to generate back-translations, which are then used to further train the translation model, and so on. We also incorporate an iterative training paradigm with back-translation into our zero-resource NMT model described in chapter 4, but do not see meaningful improvements with multiple iterations.

These and subsequent unsupervised NMT models have been remarkably successful at translation without parallel data. However, they are unable to translate well when one of the languages is morphologically rich or when the monolingual corpora are dissimilar (Guzmán et al., 2019). In addition, it would be ideal to combine such methods with other types of data (including parallel corpora, multilingual corpora, or linguistic annotations) when such data is available. Thus, it is still interesting to explore supervised and semi-supervised methods for low-resource neural machine translation, as we do in this thesis.

2.4 Multilingual Data in Neural Machine Translation

Machine translation is usually trained using only data from the two languages of interest (the source language and the target language). However, corpora in other languages have proven to be helpful in improving neural machine translation in many scenarios. Two languages may have little parallel data between them but large amounts of parallel data with a third language (e.g. there is little German↔Chinese parallel data but abundant German↔English and Chinese↔English parallel data). In this case, it can be beneficial to perform pivot-based translation (section 2.4.1) or zero-shot/zero-resource translation (section 2.4.3). Finally, there has been some research showing that it can be helpful to combine parallel corpora in several languages to create multilingual neural machine translation systems (section 2.4.2). Chapter 4 of this thesis relates to using multilingual data in NMT, specifically to using monolingual data in a third pivot language to improve zero-resource NMT performance.

2.4.1 Neural Machine Translation Using Pivot Languages

Pivot-based machine translation methods can be helpful when we would like to translate between two languages that have insufficient parallel data between them. It may be the case that some parallel data with a third *pivot* language (often English) can be found; we consider this exact experimental setup in chapter 4. Pivot-based machine translation is trained on both source $\rightarrow$ pivot data and pivot $\rightarrow$ target data, but little to no direct source $\rightarrow$ target data. Pivoting can be an important technique even when translating between two high-resource languages, as those languages may not have much direct parallel data but may have plenty of parallel data through the pivot language.

A natural baseline for pivot-based machine translation would be to train a source $\rightarrow$ pivot and a pivot $\rightarrow$ target system and then simply concatenate them: first translate a source sentence to the pivot language using the first system, then translate the pivot sentence to the target language using the second system. However, this has the potential to propagate errors from the source $\rightarrow$ pivot translation into the target language. In addition, information from the source language that is relevant to translation into the target language may be lost in the pivot language. For example, grammatical gender and case distinctions would be lost if the pivot language is English. Another issue in pivot-based NMT is the two-step inference process, which is more time-consuming than direct source $\rightarrow$ target translation.

Cheng et al. (2017) propose joint training for pivot-based NMT in order to remedy some of these problems. They do this by modifying the NMT training objective to combine source $\rightarrow$ pivot and pivot $\rightarrow$ target translation into a single objective with a connection term that encourages the model to connect the two systems. However, the resulting model still has the issue that it requires two-step inference (source $\rightarrow$ pivot and pivot $\rightarrow$ target). The zero-resource methods described in section 2.4.3 attempt to solve this issue by fine-tuning on artificial direct data to allow for direct source $\rightarrow$ target inference; we present a zero-resource NMT model that is successful without two-step inference in chapter 4.

2.4.2 Multilingual Neural Machine Translation

Even when we have a parallel corpus between the desired source language and target language, multilingual data can still be helpful in neural machine translation. In this section, we discuss the task of multilingual neural machine translation, in which parallel corpora in multiple languages are used to train a single NMT system. This technique has the potential to improve machine translation between low-resource languages through transfer learning; in addition, using a single model to translate multiple languages can be beneficial because it reduces the number of models needed (Johnson et al., 2017). We use multilingual neural machine translation techniques as the basis for our zero-resource models in chapter 4.

Multilingual NMT was first considered by Dong et al. (2015) for the case of one-to-many multilingual translation, i.e. translating from a single source language into several target languages. They use a multi-task framework with a single encoder for their source language and a separate decoder and attention mechanism for each of their target languages. However, the fact that this model requires a separate decoder and attention mechanism for all target languages means that extending to more target languages or to multiple source languages could potentially be costly. Firat et al. (2016a) expand on this to consider many-to-many translation; they use several encoders (one for each source language), several decoders (one for each target language), and a single shared attention mechanism for all language pairs. The multilingual models outperform single models in low-resource cases; for high-resource cases the multilingual models are slightly worse for translating out of English but better for translating into English (possibly because all of their training data was parallel with English). Both of these models show that multilingual NMT trained in a multi-task framework can be helpful for NMT performance, but both architectures require very large models with several encoders and/or decoders.

Ha et al. (2016) and Johnson et al. (2017) simplify the multilingual NMT architecture by using a single encoder for all source languages and a single decoder for all target languages. This is achieved by appending a tag to the source sentence to indicate the desired target language (based on the method of Sennrich et al., 2016a for controlling politeness). The model is then trained on multilingual data and learns to translate from any of the source languages into the correct target language. We use the model of Johnson et al. (2017) as the basis for our work in chapter 4, and also as an inspiration for our multi-task model in chapter 6 (where instead of the tasks being translating into different languages, they are translation and parsing).

2.4.3 Zero-Shot and Zero-Resource NMT

Chapter 4 focuses on the tasks of zero-shot and zero-resource neural machine translation, which we describe in more detail here. Zero-shot neural machine translation, i.e., NMT between two languages for which no parallel data was used at training time, can be achieved by leveraging the multilingual NMT systems described in section 2.4.2. Firat et al. (2016b) first attempted zero-shot NMT with their multilingual model consisting of several encoders and decoders, but found that without fine-tuning, the model is not able to translate between the zero-shot language pairs. On the other hand, multilingual NMT with a shared encoder and decoder (Ha et al., 2016; Johnson et al., 2017) is more successful at zero-shot NMT, although its performance still lags behind that of pivoting. In chapter 4, we use both zero-shot and pivoted multilingual NMT systems as our baselines and improve over them using monolingual pivot-language data.

Zero-resource translation can be used to improve direct translation performance in zero-shot neural machine translation systems. Zero-resource NMT starts from a multilingual NMT system and improves the zero-shot direction (the direction of interest, for which no direct parallel training data is available) using pseudo-parallel corpora. These pseudo-parallel corpora are generally created from the original parallel corpora. Firat et al. (2016b) found that zero-shot NMT performance could be strongly improved using zero-resource NMT as follows:

1. Take the pivot side of the target $\leftrightarrow$ pivot parallel data and use the multilingual NMT system to back-translate it into the source language.

2. Combine the resulting machine-translated source data with the target side of the target $\leftrightarrow$ pivot parallel corpus to create a source' $\rightarrow$ target pseudo-parallel corpus.

3. Fine-tune the multilingual NMT system on the source' $\rightarrow$ target pseudo-parallel corpus.

We replicate these results for the multilingual NMT model of Johnson et al. (2017) in section 4.4.1. Because of this model's success in direct translation, we build off of it to create our zero-resource systems in chapter 4. Lakew et al. (2017) use a similar zero-resource technique to improve the multilingual NMT system of Johnson et al. (2017).

Their approach differs from that of Firat et al. (2016b) in that they back-translate directly from the target language to the source language when creating the source' $\rightarrow$ target pseudo-parallel corpus, instead of translating from the pivot language. They show that their model is successful for low-resource NMT; however, in our experiments in section 4.4.1, we find that the approach of Firat et al. (2016b) works better for our high-resource task. Finally, Park et al. (2017) combine both of these methods and also include NMT-generated sentences on the target side of the pseudo-parallel corpora. Our work in chapter 4 similarly uses machine-generated sentences on the target side as well as the source side, but we make the additional contribution that we use monolingual data in the pivot language for this task.

2.5 NMT with Syntax and Structure

As described in section 2.2.6, neural machine translation in its standard form (trained only on parallel data) makes some errors in modeling syntax. This may create problems in machine translation fluency (particularly target syntax) and adequacy (particularly source syntax). Furthermore, this problem may be exacerbated for low-resource scenarios, where the NMT model might not have enough data to infer even basic grammatical rules. Thus, a popular topic in recent years has been injecting syntax into neural machine translation; we review some such work in this section. In chapters 5 and 6, we consider the task of adding source syntax to RNN-based and transformer-based NMT, respectively, while in chapter 7, we inject unsupervised structure into the NMT encoder.

2.5.1 NMT with Source Syntax Using a Modified Encoder

Chapters 5 and 6 of this thesis focus on injecting source-side syntax into NMT. Both chapters build on previous work that incorporates source syntax into RNN-based NMT by modifying the encoder architecture; we review some of this previous work in this section.

Eriguchi et al. (2016) augment the RNN encoder with a tree-LSTM (Tai et al., 2015) to read in source-side parses. They combine this with a standard RNN decoder, the only modification to the decoder being that it can attend to phrase nodes as well as words. This model uses an external automatic parser to parse the source sentences of the parallel training data. Zaremoodi and Haffari (2018) expand on this tree-to-sequence model by introducing a forest-to-sequence architecture that is able to capture uncertainty by modeling multiple possible parses of the source sentence. Similarly, Bastings et al. (2017) replace the RNN encoder with a graph convolutional network (GCN) encoder. The GCN is able to encode graph-structured data, so source sentences can be represented using their dependency parses.

These models improve over non-syntactic RNN-based NMT systems, showing that source-side syntactic annotations can indeed help NMT performance. Thus, we consider adding source syntax into NMT to be a promising area for research. However, the models discussed in this section rely heavily on parsed data during both training and inference. In chapters 5 and 6, we introduce models that take advantage of source-side syntactic parses but are also able to deal with unparsed data at inference time; in chapter 7, we present an architecture that uses a tree structure to model the source sentences without requiring parsed training data. In addition, when the architecture of the encoder is changed entirely, as in these models, it is not clear how to incorporate the models into newly proposed architectures. For example, the transformer has improved over LSTMs in many cases, but it may be hard to apply the improvements from using a tree-LSTM or GCN encoder to the transformer architecture. Our models proposed in chapters 5 and 6 attempt to address this issue by being architecture-agnostic.

2.5.2 Linearized Parses in NMT

Our work in chapters 5 and 6 relies on linearized parses of the source sentences to inject source syntax into NMT in a flexible way that allows for different architectures and for unparsed sentences. In using linearized parses, we build off a large body of related work in syntactic NMT, which we review in this section.

Luong et al. (2016) use a single encoder and different decoders to train two tasks: parsing the source sentence and translating from source to target. Kiperwasser and Ballesteros (2018) also apply multi-task learning to syntactic NMT; they use a shared RNN decoder for translation, dependency parsing, and part-of-speech tagging and evaluate different scheduling techniques to combine the tasks. Our multi-task system presented in chapter 6 builds off these two papers by training a joint NMT and constituency parsing model using a single encoder and decoder in a transformer framework, and we further evaluate the multi-task framework on several diverse language pairs, rather than just on English↔German.

The mixed RNN encoder model of Li et al. (2017) is similar to the shared encoder model that we introduce in chapters 5 and 6. Their model uses an RNN to encode a linearized parse of a source sentence, but attends only to the words of the parse, whereas our shared encoder model is trained on both linearized parses and unparsed sentences, but for the linearized parses we attend to words and to parse labels. Both models are flexible enough to be used in different architectures, but the shared encoder model has the advantage that it is able to translate just as well from unparsed as from parsed source sentences.

In this thesis, we concentrate on source-side syntax, but linearized parses have also been popular for incorporating target syntax into neural machine translation. Aharoni and Goldberg (2017) and Nadejde et al. (2017) both train RNN-based neural machine translation systems to translate from sequential source sentences into linearized parses of target sentences. These methods have the advantage that no modifications to the decoder architecture are necessary. These models are similar to the models we propose for using source syntax in chapters 5 and 6, particularly to the shared encoder and multi-task models.

2.5.3 NMT with Unsupervised and Semi-Supervised Structure

So far, we have discussed improving neural machine translation with explicit syntactic annotation. Syntactic NMT models are often successful at improving NMT, but most require a syntactic parser in order to create the parsed parallel data. Thus, it would not be possible to apply most syntactic NMT models to very low-resource scenarios. In chapter 7, we address this challenge by inducing a tree structure over the source sentences in an unsupervised manner. Here, we review some other models that use unsupervised or semi-supervised structure in NMT.

Kim et al. (2017) introduce structured attention networks. These models extend the basic attention mechanism by allowing it to attend to latent structures such as subtrees. They evaluate their structured attention network on character-level and word-level NMT, but their results are a bit inconsistent: structured attention does not make a large difference for word-level NMT but yields improvements over standard attention for character-based NMT. Our unsupervised tree2seq model (chapter 7) induces the structure in the encoder, rather than in attention; it would be interesting to see whether these two methods could be combined effectively.

Hashimoto and Tsuruoka (2017) add a latent graph parser to the NMT encoder, allowing it to learn dependency-like soft parses in an unsupervised manner. Their fully unsupervised model does not yield meaningful improvements over a standard seq2seq model. However, pre-training the parser with a small amount of gold dependency parse annotations results in better performance than non-syntactic NMT. Our unsupervised tree2seq model in chapter 7 also learns unsupervised parse-like structures, although our model generates discrete parsing decisions and improves over seq2seq in low-resource cases without requiring any syntactic pre-training.

2.6 Conclusions

In this chapter, we have given an overview of neural approaches to machine translation, as well as several methods for using non-parallel data in neural machine translation. In the remainder of the thesis, we discuss our contributions to the goal of incorporating non-parallel data into NMT, focusing on monolingual data from the target language (chapter 3) and from other languages (chapter 4), as well as on syntactic annotations (chapters 5 and 6) and hierarchical structure (chapter 7).

Chapter 3

Augmenting Neural Machine Translation with Copied Monolingual Data

This chapter addresses the goal of augmenting the parallel neural machine translation training data with a monolingual corpus in the target language. This chapter is based on Currey et al. (2017). The technique introduced in this chapter was used in the best constrained English↔Turkish NMT systems from the WMT17 news translation shared task (Bojar et al., 2017; Sennrich et al., 2017a).

3.1 Introduction

Target-language monolingual data is one of the most popular forms of non-parallel data used in neural machine translation, partly because it is readily available for many language pairs. In section 2.3, we discussed several methods for incorporating target-language monolingual data into neural machine translation. Two effective such methods are back-translation (Sennrich et al., 2016c) and adding a language model (Gulcehre et al., 2015). Both methods have proven to be effective at adding monolingual data to standard NMT, but they suffer from some issues. First, they can be slow, since they both require pre-training: back-translation trains a target $\rightarrow$ source NMT model and translates all of the monolingual data, while language modeling methods require a language model to be trained separately. Second, back-translation in particular generally leads to larger improvements in NMT performance but depends on the quality of the back-translations (Sennrich et al., 2016c), which is partially dependent on the quality of the parallel data used to create the reverse system. In a low-resource scenario, the small amount of parallel data might hinder the improvements that can be made through back-translation. Our goal in this chapter is to develop a method for semi-supervised neural machine translation that does not suffer from these issues. Our approach uses a copied parallel corpus, i.e. one in which the target and source sentences are identical, to augment the parallel data.

We focus on language pairs with small amounts of parallel data, since we expect monolingual data to have the most impact on such cases. On the relatively low-resource language pairs of English$\leftrightarrow$Turkish and English$\leftrightarrow$Romanian, we find that our copying technique is effective both alone and combined with back-translation. This is the case even when no additional monolingual data is used (i.e. when the copied corpus and the back-translated corpus are identical on the target side).

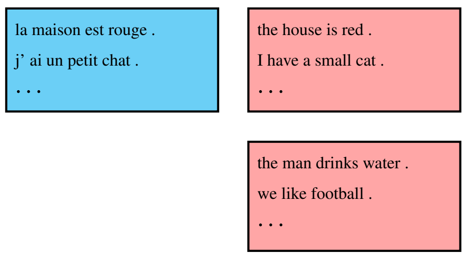
In this chapter, we introduce a straightforward and effective model for incorporating target-language monolingual data into neural machine translation. This method converts a monolingual target-language corpus into a parallel corpus by copying it, so that each source sentence is identical to its corresponding target sentence; the copied corpus is then shuffled together with the original parallel data and used to train the NMT system. The three main advantages of this method are:

- It requires no pre-training (of a reverse machine translation system or a language model), making it faster than both language modeling-based methods and back-translation.
- It does not depend on the parallel data, so it can be successful even when relatively little parallel data is available.
- It is flexible: it can easily be combined with other methods (such as back-translation) and applied to new model architectures.

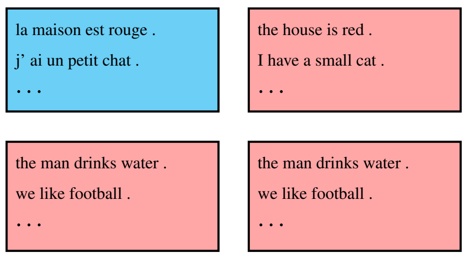
3.2 Copied Monolingual Data

We propose a method for incorporating monolingual target-side data into low-resource neural machine translation that does not rely heavily on the amount or quality of the parallel data. Throughout this chapter, we refer to our proposal as the *copied corpus* method. Figure 3.1 gives an overview of how this method words.

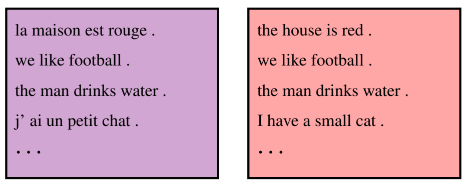
We first convert the target-language monolingual corpus into a bitext by making each source sentence identical to its target sentence; i.e., the source side of the bitext is



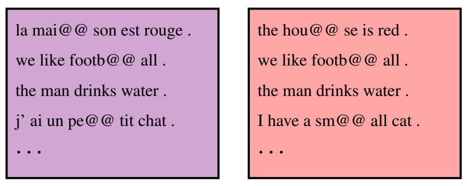
(a) Start with a parallel corpus and a monolingual target corpus.



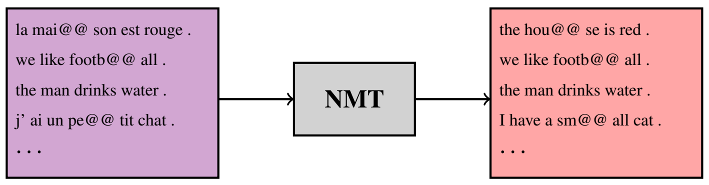
(b) Copy the monolingual corpus to convert it into a bitext.



(c) Shuffle the parallel corpus and copied corpus together.



(d) Use BPE to represent all data in the same vocabulary.



(e) Train the NMT system like normal.

Figure 3.1: Process for training an NMT system using the copied corpus method. For simplicity, we omit using back-translated data from this illustration.

a copy of the target side. We refer to this bitext as the *copied corpus*. The copied corpus is then mixed with the bilingual parallel corpus and no distinction is made between the two corpora. Finally, we train our NMT system with a single encoder and decoder using this mixed data. Since both source and target data is represented using identical byte pair encoding vocabularies trained on the concatenation of the source and target parallel data, we are able to use the same encoder for both the parallel and copied source sentences.

This copied corpus method can also be combined with the back-translation method of Sennrich et al. (2016c). This is done by shuffling the parallel, back-translated, and copied corpora together into a single dataset and training the NMT system like normal, again making no distinction between the three corpora during training. In the main experiments (section 3.4.1), we use the same monolingual data as the basis for both the back-translated and the copied corpora (so the target sides of the two corpora are identical). This means that each sentence in the original monolingual corpus occurs twice in the training data.

3.3 Experimental Setup

3.3.1 Data

We evaluate our proposed copied corpus method using three language pairs: English (EN)↔Turkish (TR), EN↔Romanian (RO), and EN↔German (DE). These language pairs were chosen because they represent low-, medium-, and high-resource scenarios, respectively, with regard to the amount of parallel data. Each language used (EN, TR, RO, and DE) has a large amount of high-quality monolingual data available.

The EN↔RO data comes from the WMT16 news translation shared task (Bojar et al., 2016), while the EN↔DE and EN↔TR datasets come from the WMT17 news translation shared task (Bojar et al., 2017). For each language pair, we use all available parallel data from the task. Validation is done on newsdev2016 for EN↔RO and EN↔TR, and on newstest2015 for EN↔DE. We use newstest2016 as the test set for all language pairs and additionally evaluate on newstest2017 for EN↔TR and EN↔DE.

The EN↔RO monolingual data is randomly sampled from News Crawl 2015, and the EN↔TR and EN↔DE monolingual data is randomly sampled from News Crawl 2016. In addition to our copying method, we back-translate the monolingual data to create a pseudo-parallel corpus with an artificial source as described in Sennrich et al.

Language Pair	Parallel	Monolingual
EN↔TR	207 373	414 746
EN↔RO	608 320	608 320
EN↔DE	5 852 458	10 000 000

Table 3.1: Number of parallel and monolingual training sentences for each language pair.

(2016c). We use the same monolingual data for both copying and back-translation, unless otherwise noted. Table 3.1 displays the amount of parallel and monolingual training data used for each language pair.

For all language pairs, we use the Moses tokenizer and truecaser (Koehn et al., 2007) to tokenize and truecase the data. We remove training sentences with more than 80 words, as well as empty sentences. Finally, we use byte pair encoding to break words into subword units (Sennrich et al., 2016d), with 89,500 BPE operations. The BPE vocabularies are trained on the concatenation of the source and target parallel data, without using the monolingual data. For RO→EN, we remove diacritics from the source side of the data, following Sennrich et al. (2016b).

3.3.2 Implementation and Training

We train attentional sequence-to-sequence models (Bahdanau et al., 2015) with a GRU encoder and decoder (Cho et al., 2014b). All models are implemented in Nematus (Sennrich et al., 2017b) following the hyperparameter recommendations of Sennrich et al. (2016b). We use hidden layers of size 1024 and word embeddings of size 512. The models are trained using Adam (Kingma and Ba, 2015) with a minibatch size of 80 and a maximum sentence length of 50. We apply dropout (Gal and Ghahramani, 2016) in all of our EN↔TR and EN↔RO systems with a probability of 0.1 on word layers and 0.2 on all other layers. Since EN↔DE is a high-resource language pair, there is less of a risk of overfitting, so we do not use dropout for EN↔DE. For all models, we use early stopping based on perplexity on the validation dataset. We decode using beam search on a single model with a beam size of 12, except for EN↔DE, where we use a beam size of 5 (which showed best results in preliminary experiments). For the experiments which use back-translated versions of the monolingual data, the target → source systems used to create the back-translations have the same setup as those used in the final source → target experiments.

	EN→TR		**TR→EN**		**EN→RO**	**RO→EN**	**EN→DE**		**DE→EN**	
BLEU	**2016**	**2017**	**2016**	**2017**	**2016**	**2016**	**2016**	**2017**	**2016**	**2017**
baseline	12.8	14.2	18.5	18.3	23.8	34.5	**33.3**	**26.6**	40.1	33.8
copied	**14.0**	**15.2**	**18.9**	**18.6**	**24.5**	**35.7**	**33.3**	26.3	**40.2**	**34.0**

Table 3.2: Translation performance in BLEU with and without copied monolingual data on the newstest2016 and newstest2017 datasets.

3.3.3 Baseline

As a baseline, we use a standard seq2seq neural machine translation model identical to the one described in section 3.3.2. This model is trained using the exact same parallel and monolingual data as our proposed copied model; the only difference is how the monolingual data is used. In the baseline, we use only the back-translated version of the monolingual target data, without any copied corpus.

3.4 Results

3.4.1 Main Experiments

This section contains our main results comparing the proposed copied corpus method to the baseline. We first evaluate the translation performance, then further investigate possible explanations for these BLEU results.

Translation Performance

Table 3.2 displays the translation quality for each language pair and each system, as approximated by BLEU score (Papineni et al., 2002). We report case-sensitive detokenized BLEU calculated using `mteval-v13a.pl`. The only difference between the baseline and the copied systems is the use of the copied monolingual corpus; all systems include back-translated data.

We observe improvements of up to 1.2 BLEU when adding a copied corpus for the low-resource (EN↔TR) and medium-resource (EN↔RO) language pairs. This indicates that our copied corpus method improves neural machine translation performance in cases where only a moderate amount of parallel data is available. For EN↔DE, we do not see any improvements from adding the copied corpus. We conjecture that this is because EN↔DE is a high-resource language pair. However, the EN↔DE sys-

Perplexity	**EN→TR**	**TR→EN**	**EN→RO**	**RO→EN**	**EN→DE**	**DE→EN**
reference	700.0	146.7	202.4	118.1	231.0	116.5
baseline	**921.1**	**341.6**	**328.2**	248.4	**490.6**	317.3
copied	921.6	344.2	344.8	**245.5**	493.3	**314.2**

Table 3.3: Language model perplexities for the outputs of each NMT system.

tems trained with the copied corpus also do not perform significantly worse than those without.

Fluency

Adding the target-side copied monolingual corpus to the training data results in improvements in translation performance as measured by BLEU score for EN↔TR and EN↔RO, even over models that use the same monolingual data with only a back-translated source. In this section and the next section, we further investigate the outputs of each system in order to better understand the source of these BLEU gains.

One possible explanation for the improvements is that the incorporation of additional target-side data helps the system generate more fluent outputs. This would be analogous to statistical machine translation, where a monolingual target corpus can be used to improve the language model.

In order to evaluate this assumption, we train a language model for each target language and use the trained language models to evaluate the perplexity of the translation outputs. We use 5-gram language models trained using KenLM (Heafield, 2011) on the full monolingual News Crawl 2015 and 2016 datasets. The data is preprocessed as described in section 3.3.1, although no subword segmentation is used.

We compare perplexities of the baseline system outputs, the outputs of the proposed system using the copied corpus, and the reference translations. For EN↔RO, we use the newstest2016 output, while for EN↔DE and EN↔TR, we concatenate the newstest2016 and newstest2017 sets into a single dataset before finding the perplexity. The results for these experiments are shown in Table 3.3. For perplexity, lower is better.

As expected, perplexities for the reference translations are significantly lower than perplexities for the translation outputs (both of the baselines and of the copied corpus systems). However, perplexities for the baselines and for the proposed copied corpus systems are similar for all language pairs, and improvements in BLEU (see Table 3.2)

Accuracy	**EN→TR**	**TR→EN**	**EN→RO**	**RO→EN**	**EN→DE**	**DE→EN**
baseline	77.3%	85.0%	71.5%	85.3%	78.5%	**91.4%**
copied	**82.0%**	**89.1%**	**78.5%**	**91.5%**	**78.6%**	91.1%

Table 3.4: Copying accuracy for the outputs of each NMT system.

do not correlate to improvements in perplexity. These results indicate that the BLEU improvements in the copied corpus systems do not seem to be due to improvements in fluency, at least as approximated by language model perplexity.

Copying Accuracy

Another possible explanation for the improvements in BLEU reported in Table 3.2 is that the copied corpus systems are better able to copy relevant words (such as numbers or named entities) to the output when appropriate. We test this hypothesis as follows:

1. For each sentence in the tokenized test data, we detect *copied words*: words that occur in both the source and the reference (ignoring case and subword segmentation).
2. We exclude words that only consist of one character.
3. We then count how many copied tokens occur in the corresponding sentence in the translation output of each system.
4. Finally, we calculate *copying accuracy* as the percent of copied tokens that occur in the corresponding sentence of the output.

The copying accuracy for each system is displayed in Table 3.4. These results mirror the BLEU results: for all language pairs except EN↔DE, there is a large improvement in copying accuracy for our proposed copied corpus systems. These results suggest that the copied corpus is able to train the model to pass appropriate words through to the target output more successfully. Table 3.5 shows some examples of translations with improved copying accuracy for the systems trained with the copied corpus.

3.4.2 Additional English-Turkish Experiments

In addition to the main evaluation described in section 3.4.1, we perform several experiments to further probe the behavior of our proposed method. We run these experiments

	RO→EN
source	... analist șef în cadrul **Peterson** Institute for International Economics.
reference	... senior fellow at the **Peterson** Institute for International Economics.
baseline	... chief analyst at the **Carson** Institute for International Economics.
copied	... chief analyst at **Peterson** Institute for International Economics.
source	Les **Dissonances** a aparut pe scena muzicala n 2004 ...
reference	Les **Dissonances** appeared on the music scene in 2004 ...
baseline	Les **Dissonville** appeared on the music scene in 2004 ...
copied	Les **Dissonances** appeared on the music scene in 2004 ...
	TR→EN
source	**Metcash**, **Aldi** istilasıyla mücadele etmek için halk kampanyası başlattı
ref	**Metcash** launches grassroots campaign to fight **Aldi** incursion
base	**Mette** launches public campaign to fight **Maldi** isticula
copied	**Metcash** launches public campaign to fight **aldi** sisters
source	PSV teknik direktörü Phillip **Cocu**, şöyle dedi: “Çok kötü bir sakatlanma.”
reference	Phillip **Cocu**, the PSV coach, said: “It’s a very bad injury.”
baseline	PSV coach Phillip **Coker** said: “It was a very bad injury.
copied	PSV coach Phillip **Cocu** said: “It’s a very bad injury.”

Table 3.5: Comparison of translations generated by baseline and copied corpus systems.

BLEU	**2016**
parallel + back-translated	12.8
parallel + double back-translated	13.1
parallel + back-translated + copied	**14.0**

Table 3.6: EN→TR translation performance when using the back-translated corpus twice vs. the back-translated and copied corpora.

on the EN→TR data described in section 3.3.1 and evaluate them on the newstest2016 test set. We choose EN→TR as our test case for further analysis since this was the language pair where the copied corpus yielded the largest improvements. It should be noted that these experiments are not directly comparable with the ones in the previous sections, because many of them use different monolingual corpora (e.g. of different sizes).

Double Back-Translated Data

In our main experiments, we used each sentence from the monolingual corpus twice in the training data: once with a back-translated source and once with a copied source. However, the monolingual data that we used for these experiments is of high quality and is from the same domain as the test data. Therefore, it is possible that the improvements shown in the main experiments are simply due to using this high-quality data twice (in the form of back-translated and copied data), rather than to using the copied corpus itself.

In order to evaluate this possibility, we consider an alternative configuration that uses the monolingual data twice without copying it. This system is trained on two instances of the same back-translated data as well as on the parallel data. Thus, the target side of this data is identical to the data used to train our proposed copied corpus systems.

Table 3.6 displays the results for these experiments. *Parallel + back-translated* is the baseline that uses a single instance of the back-translated data, *parallel + double back-translated* is the novel configuration using two instances of the same back-translated data, and *parallel + back-translated + copied* is our proposed copied corpus system.

The system that uses the copied corpus performs better than the other two systems by about one BLEU point. In addition, doubling the back-translated data does not

BLEU	**2016**
baseline	12.4
same copied	**13.6**
different copied	13.3

Table 3.7: EN→TR translation performance when using the same or different data for copied and back-translated corpora.

result in any meaningful improvement over the baseline. Thus, this indicates that the improvements in our proposed copied corpus system are not simply due to the higher weight given to the high-quality monolingual data.

Different Copied Data

Throughout the experiments presented in this chapter, we have used the same monolingual data to create both the back-translated and the copied bitexts; thus, each monolingual target sentence appears twice in the training data. An open question is whether the same gains are seen when different data is used for back-translation and for copying.

We address this question by randomly splitting the target monolingual corpus in half; we then back-translate one half of the corpus and copy the other half to create the pseudo-parallel data. Therefore, in these experiments the monolingual corpus is the same size as in the main experiments; however, the final training corpus contains each monolingual sentence only once (rather than twice as in the original copied corpus systems). We refer to this modified method as the *different copied* method.

Table 3.7 shows the results for using the different copied method, as well as the baseline (without copied data) and our original (*same copied*) proposed method. Both copied corpus systems outperform the baseline, as seen in other experiments, with the different copied version doing slightly worse than the original version.

Copied Corpus Without Back-Translation

The main results in section 3.4.1 show that our proposed copied corpus method stacks with back-translation to improve NMT quality for low- and medium-resource scenarios, even when no additional monolingual data is used. However, it is worthwhile to find out whether the copied corpus method can be used by itself, without any back-translated data. This can be helpful in cases where back-translated data is not available,

BLEU	**2016**
parallel only	9.4
parallel + small back-translated	**12.0**
parallel + large back-translated	**12.4**
parallel + small copied	11.7
parallel + large copied	12.0

Table 3.8: EN$\rightarrow$TR translation performance without back-translated data.

for example because of time constraints that make it infeasible to train a backwards system and back-translate all of the monolingual data.

The experiments examining this question are identical to the main experiments, except that no back-translated version of the data is used; only parallel and copied monolingual data make up the training corpus. We consider two scenarios: using a small and large copied corpus. The large corpus contains roughly 400k sentences, as in the main experiments, whereas the small corpus contain a randomly selected 200k-sentence subset of the large corpus.

The results for these experiments are shown in Table 3.8. The first row of the table shows the baseline system trained with parallel data only, and we include systems trained with parallel and back-translated data (without copied data) for comparison (rows two and three). In rows four and five, we display the results for the copied corpus methods without back-translation. Both the small and the large copied corpora yield strong improvements over the parallel-only baseline (2.3–2.6 BLEU), and their performance is only slightly worse than that of the corresponding back-translation systems (0.3–0.4 BLEU). Thus, using copied data without back-translation is a relatively quick and effective way to incorporate target monolingual training data, although results are best when both back-translated and copied corpora are used.

Source Copied Corpus

The main focus of this chapter has been adding a target-side monolingual corpus to the training data. However, our copied corpus technique, when stacked with back-translation, can also be used as a method for adding monolingual training data in the source language. Source-side monolingual training data has the potential to aid the encoder in better understanding the source language, possibly resulting in improved translation performance.

BLEU	2016
baseline	12.4
copied	**13.6**
+ EN data	**13.6**

Table 3.9: EN→TR translation performance with EN monolingual data.

We propose incorporating source monolingual data as follows:

1. Create a copied corpus from the source monolingual corpus by duplicating it.
2. Mix the copied source corpus with the parallel data.
3. Train a target → source neural machine translation system using the mixed parallel and copied data.
4. Use the target → source system to back-translate the target monolingual corpus.
5. Train the final source → target system on the parallel, back-translated, and copied target data.

Thus, the source copied corpus is used to improve the back-translations of the target monolingual data. This is similar to the iterative, bidirectional process used subsequently in unsupervised machine translation (described in more detail in section 2.3.4).

For these experiments, we use English monolingual data, in additional to the target (Turkish) monolingual data. The EN monolingual data is randomly sampled from the News Crawl 2015 corpus and preprocessed in the same way as for the parallel data (described in section 3.3.1). We use 400k monolingual EN sentences, corresponding to twice the size of the parallel corpus (and the same size as the target monolingual corpus).

Table 3.9 displays the results for the source copied experiments. Adding in the source (EN) monolingual data does not result in any improvements over the target-only copied model, although both copied systems do improve over the baseline. Thus, using copied monolingual source data to improve the back-translations does not yield gains in the final system.

Amount of Copied Data

Although we used a 2:1 ratio of copied to parallel data in the main EN→TR experiments (section 3.4.1), an open question is how effective the copied corpus method is

BLEU	**1:1**	**2:1**	**3:1**
baseline	12.0	12.4	12.8
copied	**13.0**	**13.6**	**13.8**

Table 3.10: EN→TR translation quality on newstest2016 with different amounts of monolingual data.

when the proportion of monolingual to parallel data is varied. We study this question in this section.

Three monolingual corpus sizes are considered: 200k sentences (1:1 ratio of monolingual to parallel data), 400k sentences (2:1 ratio), and 600k sentences (3:1 ratio), where each smaller monolingual corpus is a subset of the larger one. We train baseline systems using parallel and back-translated data, as well as copied corpus systems using parallel, back-translated, and copied data. In all cases the amount of back-translated data varies along with the amount of copied data. We do not oversample the parallel data to balance the data sources.

Table 3.10 displays the translation performance on newstest2016 when a 1:1, 2:1, and 3:1 ratio of monolingual to parallel data is used. The baseline BLEU scores vary due to the different amounts of back-translated data in these systems; adding more back-translated data consistently increases the BLEU score even when no more parallel data is used.

The systems trained with copied monolingual data consistently perform at least 1 BLEU better than the corresponding baselines. This is especially surprising in light of the fact that we do not oversample the parallel corpus; for the 2:1 and 3:1 cases, the overall translation performances improve despite the system seeing far less parallel than monolingual data during training. In addition, adding more monolingual data consistently yields small improvements (0.2–0.6 BLEU).

3.5 Discussion

Our proposed method of using a copied target-language monolingual corpus to augment training data for NMT proved to be beneficial for EN↔TR and EN↔RO translation, resulting in improvements of up to 1.2 BLEU over a strong baseline. We showed that our method stacks with the previously proposed back-translation method of Sennrich et al. (2016c) for these language pairs. For EN↔DE, however, there was no significant difference between systems trained with the copied corpus and those trained without it. There was much more parallel training data for EN↔DE than for EN↔RO (nearly 10 times as much) and EN↔TR (about 28 times as much), so it is possible that the gains that would have come from the copied corpus were already achieved with the parallel data. Overall, the copied monolingual corpus either helped or was indifferent, so training with this copied target corpus is not risky, although subsequent work found that training with copied source data could damage performance (Khayrallah and Koehn, 2018). In addition, the proposed copied corpus method does not require any more monolingual data besides what is used for back-translation.

We initially assumed that the copied monolingual corpus was helping to improve the fluency of the target outputs. However, further study of the outputs did not support this assumption, as noted in section 3.4.1. Our method did improve accuracy of copying words that are identical in the source and target languages; this may be part of the explanation for the increases in BLEU score when using the copied corpus.

Besides our copied corpus method, there are several other techniques that could improve copying from source to target. For instance, Luong et al. (2015b) detected source alignments to out-of-vocabulary target words and either replaced the words with their dictionary translations or copied the source words to the target as a postprocessing step. Another option would be to assign a higher weight in the objective function to words that are identical in the source and target to encourage the model to detect those words and handle them properly. Pointer-generator networks (See et al., 2017), which are trained to decide at each timestep between copying a word from the source sentence and generating a new word, could also be an effective solution for helping an NMT model learn to copy. However, there may be additional ways (besides improved copying) in which our proposed copied corpus improves translation. Evidence for this includes the experiments in section 3.4.2 that showed that using only the copied corpus did almost as well as using only back-translation. In particular, the copied corpus might help the encoder learn more universal encodings of the inputs. This could be because the copied corpus method trains a multilingual encoder (since it sees data in the source and target languages), and it combines NMT and autoencoding objectives, which has proven to be helpful in very low-resource NMT in subsequent work (Artetxe et al., 2018; Lample et al., 2018).

All of our experiments in this chapter were run on translation between languages with the same script. In principle, our proposed copied corpus method could be applied to translation between two languages with different scripts as well. However, this would require increasing the source vocabulary, since there would be little overlap between the BPE subwords of the source language and the target language. In addition, it is unlikely that the improvements in copying accuracy seen in section 3.4.1 would carry over to languages with different scripts. One solution could be to transliterate the source side of the copied corpus so that it is written in the same script as the source language. Whether transliteration is better than simple copying and whether the copied corpus method would be effective for translating between languages with different scripts remain open empirical questions.

In addition to potential issues with using the copied corpus method on translation between different scripts, the method proposed in this chapter has some other disadvantages. First, it did not result in any improvements in BLEU for high-resource EN↔DE translation. Additionally, the copied corpus method alone did not outperform back-translation alone. Finally, although we found that copied corpus method improves copying accuracy on words that should be copied from the source sentence to the output, we did not check for false positives. It is possible that the model trained with the copied corpus also learns to copy words that are not actually identical in the source and the target.

3.6 Subsequent Work

Since the publication of the research in this chapter, there have been several papers that further examined the use of copied or monolingual data in neural machine translation. Edunov et al. (2018) investigated back-translation and found that generating noisy synthetic source sentences (e.g. through sampling or noised beam search) yielded improvements over beam search- and greedy search-based back-translation, especially as more back-translated data was used. It is possible that the copied corpus benefits from the same concept, since it uses extremely noisy synthetic source sentences (which are identical to the target sentences) as well as synthetic source sentences generated from beam search. This would also fit with the fact that the copied corpus method does well even when the amount of monolingual data is three times as much as the amount of parallel data (as described in section 3.4.2).

The effect of copying source-language sentences has also been analyzed. Ott et al. (2018) discovered that in the WMT14 English↔French and English↔German datasets, between 1.1% and 2% of training sentence pairs are in fact copies. They showed that copies of the source-language sentence on the target side of the data can significantly worsen model outputs, particularly when large beam sizes are used during inference. Similarly, Khayrallah and Koehn (2018) showed that when the source training sentences are copies of the corresponding target sentences (i.e., when the source side of the data contains target-language data), only small decreases in BLEU occur; however, when the target sentences are copies of the source sentences, the model performance is strongly degraded. In fact, they found that models learned to copy the source sentence when only 20% of the target sentences were copies of the source. Thus, although we have shown that training models to copy target-language text is an effective secondary task that can improve NMT, it is likely that training them to copy source-language text would worsen NMT performance.

3.7 Conclusions

In this chapter, we introduced a method for improving neural machine translation using monolingual data, particularly for low-resource scenarios. Augmenting the training data with monolingual data in which the source side is a copy of the target side proved to be an effective way of improving EN$\leftrightarrow$TR and EN$\leftrightarrow$RO translation, while not damaging EN$\leftrightarrow$DE (high-resource) translation. This technique could be used in conjunction with back-translation but was also effective without back-translation (though not quite as strong as back-translation). In addition, using much more monolingual than parallel data did not hinder performance, which is beneficial for the common case where a large amount of monolingual data is available but the language pair has little parallel data.

The overarching goal of this thesis is to investigate ways of improving neural machine translation by using different types of training data in addition to the traditional parallel corpora. This is particularly important in low-resource scenarios, where sufficient parallel data may be unavailable. In this chapter, we have concentrated on monolingual data in the target language, introducing a copied corpus method that can improve neural machine translation significantly for low- and medium-resource language pairs; we have also briefly explored using monolingual data in the source language. In subsequent chapters, we will expand on this by using monolingual data from a third pivot language and syntactic data.

This work invites natural extensions that we would like to investigate in the future. Specific subtasks of machine translation, such as domain adaptation and translation between similar languages, could potentially be improved using a copied monolingual corpus; it would be interesting to apply the methods and experiments in this chapter to such subtasks. In addition, the quality of the monolingual data has been shown to have an effect on NMT performance (Cheng et al., 2016), so applying a data selection method when creating the copied corpus is also a promising avenue for future research.

Chapter 4

Zero-Resource NMT with Monolingual Pivot Data

In this chapter, we build on the work of the previous chapter on using monolingual data to improve neural machine translation. Here, we concentrate on pivot-based and zero-resource NMT and use monolingual data from the pivot language. A paper based on this chapter is currently in submission.

4.1 Introduction

Most modern neural machine translation methods depend heavily on the availability of parallel data between the source and the target language. In chapter 3, we considered the case of adding target-language monolingual data to a language pair that had some parallel data available. However, even two high-resource languages, such as German and Russian, may not have sufficient parallel data between them.

Although most language pairs may have little parallel data available, it is often possible to find parallel corpora with a third *pivot* language. For example, while direct German↔Russian parallel data is relatively scarce, both German↔English and Russian↔English data is abundant. Pivot-based and zero-shot NMT systems have been proposed as a means of taking advantage of this data to translate between e.g. German and Russian.

In pivot-based machine translation, text is first translated from the source language into the pivot language, and then from the pivot language into the target language. Although such methods can result in strong translation performance (Johnson et al., 2017), they have a few disadvantages. The two-step pivoting translation process doubles the latency during inference and has the potential to propagate errors from the source $\rightarrow$ pivot translation into the final target output. Additionally, there is a risk that relevant information in the source sentence can be lost in the pivot translation (e.g. case distinctions if pivoting through English) and not represented in the target sentence. Zero-shot methods that take advantage of multilingual NMT systems to perform direct source $\rightarrow$ target translation can help in addressing these problems, and zero-resource methods build off of zero-shot methods by fine-tuning them on pseudo-parallel data to improve direct translation.

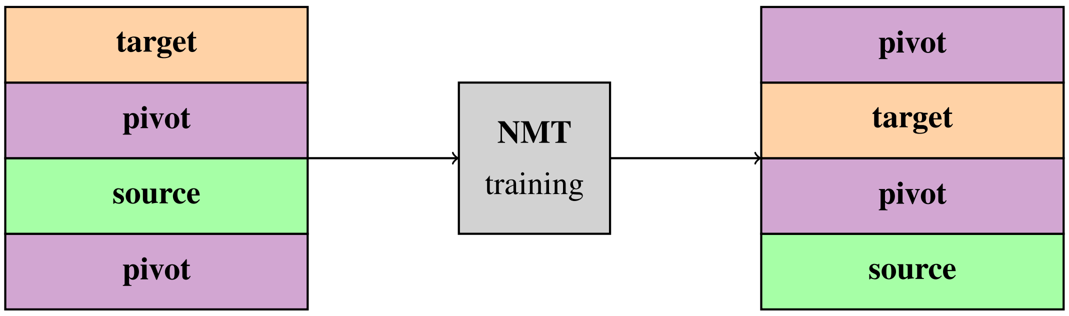
The goal of this chapter is to augment zero-resource NMT with monolingual data from the pivot language. Although there have been several explorations into using parallel corpora through a pivot language to improve NMT (Firat et al., 2016b; Lakew et al., 2017; Park et al., 2017) and using monolingual source and target corpora in NMT (Edunov et al., 2018; Gulcehre et al., 2015; Hoang et al., 2018; Niu et al., 2018; Sennrich et al., 2016c; Zhang and Zong, 2016), this is to our knowledge the first attempt at using monolingual pivot-language data to augment NMT training. Leveraging monolingual pivot-language data is worthwhile because the pivot language is often the highest-resource language of the three (e.g. English), so we expect there to be more high-quality monolingual pivot data than monolingual source or target data. Thus, we make use of parallel source $\leftrightarrow$ pivot data, parallel target $\leftrightarrow$ pivot data, and monolingual pivot-language data to build a zero-resource NMT system. Although we use a basic multilingual NMT system as the basis, the methods proposed here could easily be applied to any zero-shot NMT architecture.

4.2 Zero-Resource NMT with Pivot Monolingual Data

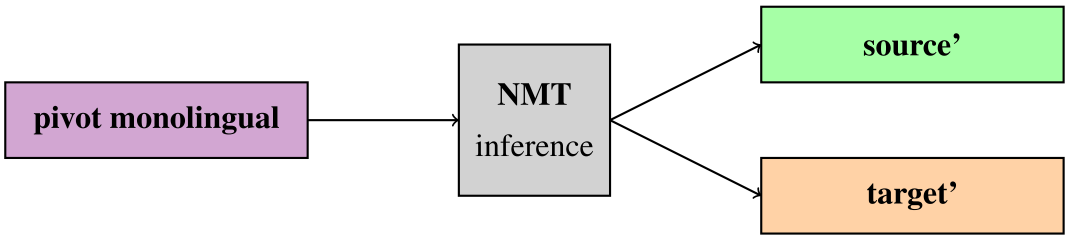
In this chapter, we concentrate on zero-resource NMT between two languages X and Y given a pivot language Z. We assume access to X$\leftrightarrow$Z and Y$\leftrightarrow$Z parallel corpora, but no direct X$\leftrightarrow$Y parallel corpus. Our goal is to use monolingual data in the pivot language Z to improve both X$\rightarrow$Y and Y$\rightarrow$X translation simultaneously. Figure 4.1 gives an overview of our proposed method.

4.2.1 Initial Multilingual Models

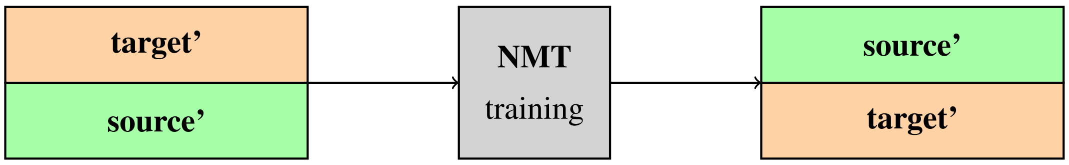
We start by reviewing the multilingual NMT models that are used as the basis for our experiments. Here, we do not consider single-directional bilingual NMT models, only



(a) An initial multilingual NMT model is trained on source $\leftrightarrow$ pivot and target $\leftrightarrow$ pivot parallel data (section 4.2.1).



(b) The pivot monolingual corpus is back-translated into the source and target languages using the trained NMT model (section 4.2.2).



(c) The source' $\rightarrow$ target' and target' $\rightarrow$ source' pseudo-parallel corpora are used to train the final NMT system from scratch or fine-tune the initial model (section 4.2.3). In practice, we concatenate this data with a subset of the original parallel data (not shown here).

Figure 4.1: Illustration of the basic steps in our zero-resource NMT model using pivot-language monolingual data.

multilingual NMT models. This is because we would like to translate directly between language X and language Y at inference time without using the pivot language; translating through the pivot language would double the amount of time it takes to translate and potentially lead to information loss or error propagation. In this work, we also do not consider the case of adding monolingual data from the main languages of interest (X and Y), although such data would likely further improve translation quality.

Our initial multilingual NMT model is based on the model introduced by Johnson et al. (2017), although here we use the transformer architecture (Vaswani et al., 2017). We train the initial model on mixed X→Z, Z→X, Y→Z, and Z→Y parallel data and use tags at the beginning and end of each source sentence to indicate the desired target language. We shuffle all of the data together randomly, regardless of source and target language. We do not employ any extensions to the zero-shot architecture (Arivazhagan et al., 2019; Lu et al., 2018; Platanios et al., 2018), although the methods described here could easily be applied to such extensions as well.

4.2.2 Back-Translation of Pivot Monolingual Data

We turn now to the task of leveraging the monolingual corpus in the pivot language Z to improve the multilingual NMT models. We aim to improve only X→Y and Y→X translation, without regard to performance on the other language pairs that are included in the multilingual system (X↔Z and Y↔Z).

We start by using the initial monolingual model described in section 4.2.1 to back-translate the monolingual pivot data into both languages of interest (X and Y). Since the initial multilingual model was trained on both these directions (Z→X and Z→Y), we expect it to do reasonably well at back-translation. Thus, for each sentence in the Z monolingual corpus, we have its translation in both X and Y, so we can create a pseudo-parallel corpus X'↔Y' (where the prime symbol indicates machine-translated text). We concatenate both directions (X'→Y' and Y'→X') together to create our back-translated pivot (*BT-pivot*) corpus. This resulting corpus contains synthetic data on both the source and the target side.

4.2.3 Using the BT-Pivot Corpus

The BT-pivot corpus uses the monolingual corpus from the pivot language Z to create a direct pseudo-parallel corpus between the two languages of interest, X and Y. In this section, we introduce three methods for using this BT-pivot data to create a zero-resource NMT system for X↔Y translation. In all cases, we concatenate the BT-pivot corpus with a subset of the original training data to train the zero-resource models; in preliminary experiments, we found that using some original training data yielded slightly higher BLEU scores than training on back-translated data alone. We take only a subset of the original parallel training data rather than the entire corpus in order to cut down on training time.

We dub our first method *pivot from scratch*. In this method, we discard the initial NMT model and train a new NMT model from scratch using the BT-pivot data (concatenated with the subset of the original parallel corpora). We use the same model hyperparameters as for the initial NMT model.

Our second method, *pivot fine-tune*, is similar to the first: both methods use the BT-pivot data (along with the subset of the original parallel data). However, for pivot fine-tune, we use the BT-pivot data and the subset of the parallel data to fine-tune the original multilingual model described in section 4.2.1, rather than training a new model from scratch.

Finally, we propose a *pivot-parallel combined* method. This method also fine-tunes the original multilingual model, but uses an augmented fine-tuning dataset. In addition to the BT-pivot corpus and the subset of the original training data, we add a back-translated parallel (*BT-parallel*) corpus generated following Firat et al. (2016b) as follows:

1. Use the initial multilingual model to translate the Z side of the subsetted X↔Z parallel corpus into language Y.

2. Combine the resulting Y' data with the X side of the subsetted X↔Z parallel corpus to create a Y'→X parallel corpus.

3. Use the initial multilingual model to translate the Z side of the subsetted Y↔Z parallel corpus into language X.

4. Combine the resulting X' data with the Y side of the subsetted Y↔Z parallel corpus to create a X'→Y parallel corpus.

5. Concatenate the two back-translated corpora (X'→Y and Y'→X) to create the BT-parallel corpus.

The BT-parallel corpus is then combined with the BT-pivot corpus and the subset of the original parallel data and used to fine-tune the initial multilingual model.

Method	Back-Translated Data	Training Regime
pivot from scratch	BT-pivot	train from scratch
pivot fine-tune	BT-pivot	fine-tune initial model
pivot-parallel combined	BT-pivot + BT-parallel	fine-tune initial model

Table 4.1: Summary of the proposed methods for zero-shot NMT using pivot-language monolingual data.

Corpus	Sentences
EN↔DE	4 497 878
EN↔RU	2 500 502
EN monolingual	1 000 000

Table 4.2: Number of sentences in each training corpus for the DE↔RU experiments.

Table 4.1 summarizes the three proposed methods for zero-shot NMT. The three methods vary in the back-translated data used (BT-pivot only vs. BT-pivot and BT-parallel) and in the training regime (training a new model from scratch vs. fine-tuning the initial multilingual model). In initial experiments, we also tried a version of the pivot-parallel combined method that trained a new model from scratch, although this did not do as well as the pivot-parallel combined method with fine-tuning.

4.3 Experimental Setup

4.3.1 Data

We run our experiments on a high-resource setting: translation between German (DE) and Russian (RU) using English (EN) as the pivot. The data comes from the WMT16 news translation task (Bojar et al., 2016). We use all available parallel corpora for EN↔DE (Europarl v7, Common Crawl, and News Commentary v11) and for EN↔RU (Common Crawl, News Commentary v11, Yandex Corpus, and Wiki Headlines) to train the initial multilingual system, but no direct DE↔RU parallel data. When the parallel data is used alongside the back-translated corpora for fine-tuning or re-training from scratch (as described in section 4.2.3), we randomly sample one million sentences from each parallel corpus.

For pivot (EN) monolingual data, we take a random subset of one million sentences from the News Crawl 2015 corpus. Since the goal of this chapter is to study the effectiveness of using pivot-language monolingual data, we do not use any DE or RU monolingual data; however, we expect that such data would also be beneficial. Table 4.2 shows the size of each training corpus after preprocessing. We use the overlapping DE and RU sentences from newstest2014 as the validation set (1505 sentences), newstest2015 as the test set (1433 sentences), and newstest2016 as the held-out set (1500 sentences). The overlapping sentences were originally written in English and were translated by human translators into German and Russian (Bojar et al., 2016).

All data is tokenized and truecased using the Moses scripts (Koehn et al., 2007). We use a joint byte pair encoding (Sennrich et al., 2016d) vocabulary for all three languages (DE, EN, and RU) trained on all parallel data with 50k merge operations. Similarly to Johnson et al. (2017), we use tags at the beginning and end of the source sentence to indicate the desired target language.

4.3.2 Implementation and Training

All models in our experiments are based on the transformer architecture (Vaswani et al., 2017). We use the Sockeye toolkit (Hieber et al., 2017) to run all experiments. We find that the default Sockeye hyperparameters work well, so we stick with those throughout. We use beam search with beam size 5 both when back-translating and during inference.

4.3.3 Baselines

Initial Models Without Monolingual Data

We compare our models to three baselines that are trained without any monolingual data. We refer to these baselines as *initial models* because they are used as the basis for our proposed models: we use them to generate the BT-pivot data and we fine-tune them using the generated data to create our proposed models.

The first baseline is a multilingual model based on Johnson et al. (2017), but we use the transformer architecture and add target language tags at both the beginning and end of the source sentences. This multilingual model is trained on the English↔German and English↔Russian parallel data. We evaluate this model both with direct (zero-shot) translation (German→Russian and Russian→German) and with pivot translation through English.

Secondly, we consider the zero-resource NMT method proposed by Lakew et al. (2017). This method consists of selecting sentences from the DE↔EN parallel corpus and back-translating them from DE into RU, resulting in a RU'→DE pseudo-parallel corpus. The same is also done with the RU↔EN parallel corpus to create a DE'→RU pseudo-parallel corpus. These corpora are then concatenated with the original parallel data and used to fine-tune the multilingual model. This zero-resource method is only evaluated on direct DE→RU and RU→DE translation (not on pivoting through EN).

We also compare our models to a zero-resource baseline based on the technique introduced by Firat et al. (2016b). This method fine-tunes the initial multilingual model with the BT-parallel corpus described in section 4.2.3 (concatenated with the origial data). Like the other zero-resource baseline, this baseline is only evaluated on direct translation (not on pivot translation).

Baselines with Monolingual Data

In addition to the initial models, we compare our proposed zero-resource NMT methods to two baselines trained with monolingual EN data. For both of these baselines, we evaluate both direct zero-shot translation and pivot translation through EN.

The first is based on our copied corpus method (chapter 3). We train an identical model to the initial multilingual model, but with additional EN→EN pseudo-parallel training data from the EN monolingual corpus. Thus, this model is trained on DE↔EN, RU↔EN, and EN→EN data. We do not fine-tune this model with any pseudo-parallel data.

The second baseline we consider is back-translation (Sennrich et al., 2016c). Starting from the trained multilingual model, we back-translate the EN monolingual data into both DE and RU, then fine-tune the multilingual model on the original training data, plus the DE'→EN and RU'→EN pseudo-parallel corpora.

4.4 Results

4.4.1 Main Experiments

Table 4.3 shows translation performance (as estimated by BLEU score) for our main experiments. We display results for initial multilingual models without any monolingual data (rows 1–4), for copied corpus and back-translation baselines using the monolingual data (rows 5–8), and for our proposed zero-resource models (rows 9–11).

		DE→RU		RU→DE	
	BLEU	**test**	**held-out**	**test**	**held-out**
initial models	multilingual direct	3.4	2.7	15.2	14.5
	multilingual pivot	21.3	19.3	21.7	20.2
	Lakew et al., 2017	19.4	17.0	14.4	13.2
	Firat et al., 2016b	22.6	20.7	21.0	18.3
baselines	copied corpus direct	3.7	3.1	10.2	9.5
	copied corpus pivot	20.9	18.9	21.1	19.9
	back-translation direct	3.7	2.9	14.8	14.1
	back-translation pivot	22.3	20.4	22.4	20.9
proposed models	pivot from scratch	23.0	20.6	22.3	21.5
	pivot fine-tune	23.0	20.3	22.4	21.5
	pivot-parallel combined	**23.6**	**21.1**	**22.5**	**21.6**

Table 4.3: BLEU scores for the initial multilingual models and zero-resource models without monolingual data, for the baselines with pivot monolingual data, and for our proposed zero-resource models with pivot monolingual data. We report results on the test set (newstest2015) and the held-out set (newstest2016). For the baselines and the initial multilingual models, we use consider both direct (zero-shot) and pivot translation.

Initial Models Without Monolingual Data

For the multilingual baseline, direct source $\rightarrow$ target translation does very poorly for DE$\rightarrow$RU. Although the performance is somewhat more reasonable for RU$\rightarrow$DE, direct translation still lags far behind pivot (source $\rightarrow$ EN $\rightarrow$ target) translation for this model. Our results differ from those of Johnson et al. (2017), who showed reasonable performance in both directions for zero-shot translation. However, they tested their zero-shot systems only on closely related languages or very large-scale multilingual systems, whereas we use somewhat smaller training sets and distantly related languages. This might be an explanation for the discrepancy in results.

Both zero-resource models (Lakew et al., 2017 and Firat et al., 2016b) outperform the multilingual baseline overall for direct translation. In addition, the latter closes the gap with the pivot translation baseline for DE$\rightarrow$RU and almost closes it for RU$\rightarrow$DE. Thus, fine-tuning on back-translated parallel data is very helpful in improving zero-resource NMT. In the next sections, we evaluate methods for further improving zero-resource NMT using EN monolingual data.

Baselines with Monolingual Data

The results for the copied corpus and back-translation baselines (using both direct and pivot translation) are shown in rows 5–8 of Table 4.3. Both models are unable to translate well using only direct translation, but when pivot translation is used, their performance improves. In particular, the back-translation pivot baseline achieves slightly higher BLEU scores overall than any of the initial models trained without monolingual data.

In chapter 3, we showed that the copied corpus method was useful for adding target-language monolingual data to NMT training. Here, we see that the same method is not beneficial (and in fact is slightly harmful compared to the baseline) for adding pivot-language monolingual data to NMT. This could be because the copied corpus is used here to improve translation directions that are not of interest (i.e. translation into and out of English, rather than DE$\leftrightarrow$RU translation).

Proposed Models with Monolingual Data

We display the results for our three proposed models in the last three rows of Table 4.3. Compared to the best pivot-based model (back-translation), the pivot from scratch and pivot fine-tune models perform slightly better overall in both translation directions

(DE→RU and RU→DE). Additionally, the pivot-parallel combined model improves over the best pivot-based model by about 1 BLEU for DE→RU and also does slightly better for RU→DE. This BLEU gain is especially interesting since the proposed models do not require two-step inference, unlike the back-translation pivot-based model.

Comparing to the best direct translation model (the zero-resource model based on Firat et al., 2016b) leads to similar conclusions. The pivot from scratch and pivot fine-tune methods do similarly to this baseline for DE→RU translation and improve over it by 1.3–3.2 BLEU for RU→DE translation. For the pivot-parallel combined model, the gains over the baseline for DE→RU are stronger than for the other two methods, and the gains for RU→DE are similar. Thus, we have shown that adding pivot-language monolingual data through these methods can improve zero-resource NMT performance.

All three of our proposed models improve over a strong direct translation baseline and perform similarly to or better than a pivot-based translation baseline that uses EN monolingual data without requiring the two-step inference process necessary for pivot-based translation. The pivot from scratch and pivot fine-tune models give similar results, while the pivot-parallel combined method, which adds in the back-translated parallel corpus, yields the best BLEU scores out of all models across the board.

4.4.2 Iterating the Proposed Models

Inspired by Hoang et al. (2018) and Niu et al. (2018), we study whether iterating the proposed models can improve translation performance. Starting from the trained proposed models from section 4.4.1, we run a second iteration as follows:

1. Back-translate the same EN data using the new model to create a new BT-pivot corpus (as described in section 4.2.2).

2. For the pivot-parallel combined method, back-translate the EN side of the parallel data as well (following Firat et al., 2016b).

3. Fine-tune the model or train the model from scratch using the new data concatenated with the subset of the original parallel data (as described in section 4.2.3).

Table 4.4 shows the performance on the test dataset (newstest2015) when a second iteration of back-translation and training is performed. For the pivot from scratch and pivot fine-tune methods, we see small gains (up to 0.4 BLEU) from running a second iteration. These small improvements help the pivot from scratch and pivot fine-tune

	DE→RU		RU→DE	
BLEU	**iter 1**	**iter 2**	**iter 1**	**iter 2**
pivot from scratch	**23.0**	**23.0**	22.3	**22.7**
pivot fine-tune	23.0	**23.3**	22.4	**22.8**
pivot-parallel combined	**23.6**	22.7	**22.5**	21.2

Table 4.4: BLEU scores for the proposed models on the test set (newstest2015). We show BLEU scores for one and two iterations (iter 1 and iter 2).

methods catch up to the single-iteration version of the pivot-parallel combined method. On the other hand, running a second iteration is very costly in terms of training time, since it requires another back-translation step and another training step. For the pivot-parallel combined model, which was the best-performing model with one iteration, adding a second iteration damages performance in terms of BLEU score. This seems to match the results of Hoang et al. (2018) that indicate that there are diminishing returns as more iterations are added.

4.5 Discussion

In this chapter, we have proposed a novel use of data for neural machine translation: using monolingual data in a pivot language to improve zero-resource NMT. We have introduced a way of generating a pseudo-parallel source ↔ target training corpus using the monolingual pivot-language corpus. We also showed three ways of leveraging this corpus to train a final source ↔ target NMT system. We compared these models to standard zero-shot and zero-resource baselines, as well as to some common-sense baselines that use monolingual data (back-translation and copied corpus).

Our proposed paradigm has several benefits. First, it shows that monolingual data from a language other than the source and target languages can aid NMT performance, complementing literature on using source- and target-language monolingual data in NMT (reviewed in detail in section 2.3). Second, this paradigm is architecture-agnostic, so it would be easy to apply to architectures that improve upon the basic zero-shot and zero-resource models (e.g. Arivazhagan et al., 2019; Lu et al., 2018; Platanios et al., 2018). Additionally, there are some advantages from an efficiency standpoint: the models proposed are successful at direct translation, which is faster than pivot-based translation. We have shown that these models can achieve BLEU gains over both direct and pivot-based baselines, even without increased inference time, so we expect that they will be easy to apply in practice.

However, the models we have proposed in this chapter are not without limitations. First, using the pivot-language monolingual data might not work as well when the source and target languages are closely related; this might be a case where source and target monolingual data is more useful than pivot monolingual data. These models also tune a multilingual NMT system for translation in two directions only (source $\rightarrow$ target and target $\rightarrow$ source), so they would not be applicable in cases where a single massively multilingual NMT system (Aharoni et al., 2019) is required. Finally, adding multiple back-translation and re-training iterations (section 4.4.2) does not result in gains in BLEU score for the best performing model.

4.6 Conclusions

This chapter introduced the task of zero-resource neural machine translation using pivot-language monolingual data. We proposed three methods for improving zero-resource NMT with monolingual pivot data: pivot from scratch, pivot fine-tune, and pivot-parallel combined. All three methods improved over strong baselines that used both direct translation and pivot translation; the pivot-parallel combined method was the most successful.

Chapter 3 explored the use of copied monolingual data in the target language for NMT; this worked well when some parallel data existed between the source and target language. In this chapter, we have expanded on the work done in chapter 3 by considering the case of parallel data only being available through a high-resource pivot language. We showed that adding monolingual data in this pivot language can strongly improve zero-resource neural machine translation. In the remaining chapters of this thesis, we will change tactics, concentrating on adding syntactic information into NMT. Both sources of outside information (monolingual data and syntactic annotations) are complementary and could be used in conjunction to improve low-resource neural machine translation.

In the future, we hope to additionally study the use of source-language and target-language monolingual data in zero-resource NMT. We would also like to test our proposed zero-resource methods on other zero-shot NMT architectures and on other language pairs. We also think that data selection methods on the back-translated data (Niu et al., 2018) could be helpful, since zero-shot multilingual NMT models often generate translations in the wrong target language (Arivazhagan et al., 2019).

Chapter 5

Multi-Source Syntactic NMT with Linearized Parses

This chapter explores a multi-source method for adding source syntactic information into neural machine translation using linearized source parses. Much of the research in this chapter was published in Currey and Heafield (2018a).

5.1 Introduction

After studying the use of monolingual data in NMT in chapters 3 and 4, we now turn our attention to leveraging syntax to improve neural machine translation performance. In this chapter, we look at adding syntax to recurrent neural network-based NMT; chapter 6 considers syntactic transformer-based NMT. The recurrent encoders and decoders often used in NMT do not explicitly model syntax, and it has been shown that RNNs do not fully learn syntax without supervision (Linzen et al., 2016). Therefore, explicitly incorporating syntactic information into RNN-based NMT has the potential to improve performance. Here, we concentrate on syntactic representations of the source sentences, which we hope will improve the model's representation of the source language.

Recently, there have been a number of proposals for adding source-side syntax into neural machine translation by updating the encoder (section 2.5.1 contains a comprehensive review). These models tend to show gains in BLEU score, but they are not without their disadvantages. Many syntactic NMT models consist of proposing a new architecture for the encoder, making them difficult to transfer to novel NMT architectures. Additionally, the models tend to be trained and evaluated using only parsed source sentences; their performance can be poor on unparsed inputs. This can be a problem for inference, since the external parser used to generate parsed data may be too slow or fail on an input sentence, especially when dealing with potentially noisy user inputs.

Using linearized representations of parsed sentences within standard neural machine translation frameworks (Aharoni and Goldberg, 2017; Li et al., 2017; Nadejde et al., 2017) has the potential to remedy the first problem. These linearized parses can inject syntactic information into NMT models without requiring significant changes to the seq2seq architecture. However, the parsed sequences used in these models are significantly longer than standard sentences (see Table 5.1 for an example), since they contain node labels in addition to words. NMT already has trouble translating long sentences, and adding source-side linearized parses could exacerbate this. Thus, it could be beneficial to use the unparsed representation of the source sentence in addition to the linearized parse.

In this chapter, we introduce a multi-source method for incorporating source syntax into neural machine translation. This model is designed to address the issues described above as follows:

- It uses linearized parses, so it can be adapted to any architecture that takes in a sequence of words.

- It has two encoders, one for parsed and one for unparsed sentences. This means that in the case that the parsed sentence is unavailable, the model can use the unparsed encoder (which encodes the standard source sentence).

- Similarly, when the parsed source sentence is very long, the model can potentially use the shorter unparsed input for information.

In addition, unlike standard non-syntactic NMT models, the multi-source syntactic NMT model can incorporate syntactic information into NMT training to improve translation performance when such information is available. We perform experiments evaluating these aspects of the proposed multi-source model, finding that it improves translation over syntactic and non-syntactic baselines. It is also able to translate from both parsed and unparsed source sentences reasonably well (and can do even better with some simple modifications), and its performance does not deteriorate as much as that of the baselines on long sentences.

5.2 Background

5.2.1 Seq2seq Neural Parsing

Neural syntactic parsing has long made use of linearized parse trees within sequential neural models (e.g. recurrent neural networks). Vinyals et al. (2015) performed constituency parsing using an attentional LSTM-based seq2seq model; they used linearized, unlexicalized parse trees on the target side and standard unparsed sentences on the source side. In addition, as in this work, they used an external parser to create synthetic parsed training data, resulting in improved parsing performance. Choe and Charniak (2016) adopted a similar strategy, using linearized parses in an RNN language modeling framework. Here, we build off of this prior work and leverage linearized parse trees in a multi-source NMT system. We reviewed related work using linearized parses in NMT in section 2.5.2.

5.2.2 Multi-Source NMT

Multi-source methods in neural machine translation were first introduced by Zoph and Knight (2016) for multilingual translation. They used one encoder per source language, and combined the resulting sentence representations before feeding them into the decoder. Firat et al. (2016a) expanded on this by creating a multilingual NMT system with multiple encoders and decoders, but did not use the multiple encoders simultaneously. Libovickỳ and Helcl (2017) applied multi-source NMT to multimodal machine translation and automatic post-editing and explored different strategies for combining attention over the two sources. In this chapter, we apply the multi-source framework to a novel task, syntactic neural machine translation.

5.3 Multi-Source Neural Machine Translation with Linearized Source Parses

We propose a multi-source method for incorporating source syntax into neural machine translation. This method makes use of linearized source parses, which we describe in section 5.3.1. Throughout this chapter, we refer to standard sentences that do not contain any explicit syntactic information as *unparsed*; see Table 5.1 for an example. Section 5.3.2 describes the multi-source method in detail.

unparsed	hi@@ story is a great teacher .
lexicalized parse	(ROOT (S (NP (NN hi@@ story)) (VP (VBZ is) (NP (DT a) (JJ great) (NN teacher))) (. .)))
unlexicalized parse	(ROOT (S (NP (NN)) (VP (VBZ) (NP (DT) (JJ) (NN))) (.)))
target	die Geschichte ist ein großartiger Lehrmeister .

Table 5.1: Example source training sentence with unparsed, lexicalized parse, and unlexicalized parse versions. We include the corresponding target sentence for reference.

5.3.1 Linearized Constituency Parses

Several different grammatical formalisms have been used to infuse syntax into neural machine translation, including dependency trees (Bastings et al., 2017; Hashimoto and Tsuruoka, 2017), CCG supertags (Nadejde et al., 2017), and constituency parses (Aharoni and Goldberg, 2017; Li et al., 2017). In this thesis, we use constituency parses to represent the syntax of the source sentences. We choose this formalism because it is easy to linearize and it has shown good results in sequence-based neural parsing (Choe and Charniak, 2016; Vinyals et al., 2015). In addition, off-the-shelf constituency parsers can achieve high accuracy for English (the source language used in our experiments) and are readily available in other high-resource languages (Manning et al., 2014). On the other hand, compared to CCG parses, linearized constituency parses are much longer; the longer input sequences can cause problems for NMT (see section 5.5.2 for further analysis). In addition, constituency parses are trees whereas dependency parses are graphs. This means that constituency parses are less flexible and potentially worse at representing languages with freer word order than English. However, to the best of our knowledge, there has been no systematic comparison of different grammatical formalisms for syntactic NMT; it remains to be seen whether the formalism chosen affects NMT performance.

Our proposed technique relies on linearized parses of the source sentences to inject source syntax into neural machine translation. Linearizing the parses allows us to add syntactic information without modifying the standard NMT encoder architecture. We generate and format the parsed data as follows:

1. In order to generate syntactically parsed training data, we use an off-the-shelf constituency parser, in this case Stanford CoreNLP (Manning et al., 2014), to parse the source side of the parallel corpus. This technique of parsing the parallel data instead of using gold parses is common in syntactic NMT (Eriguchi et al., 2016) and in neural parsing (Vinyals et al., 2015). Figure 5.1 shows an example of the resulting parse trees.

2. We linearize the resulting parses similarly to Vinyals et al. (2015) by using a depth-first tree traversal. We tokenize the opening parentheses with the node label (so that each node label begins with a parenthesis) but keep the closing parentheses separate from the words they follow. We do not use different closing parentheses for different phrase types. However, we do use a different closing parenthesis label from the one that is already in the unparsed training data. Table 5.1 shows an example of the resulting parses.

3. Following Sennrich et al. (2016d), our networks operate at the subword level using byte pair encoding with a shared vocabulary on the source and target sides. However, the parser operates at the word level. Therefore, we break words into subwords only after parsing the data. This means that a leaf may have multiple tokens without internal structure. BPE subword segmentations are represented in Table 5.1 by @@.

For our task, the parser failed on one training sentence of 5.9 million, which we discarded, and succeeded on all validation and test sentences. It took roughly 16 hours to parse the 5.9 million training sentences.

The proposed multi-source method (section 5.3.2) is trained with two variants: using *unlexicalized* parses and *lexicalized* parses. In *unlexicalized* parses, we remove the words, keeping only the node labels and the parentheses. In *lexicalized* parses, the words are included. Table 5.1 shows an example of the three source sentence formats: unparsed, lexicalized parse, and unlexicalized parse. The lexicalized parse is a combination of the unparsed version and the unlexicalized parse; as such, it is significantly longer than the other versions.

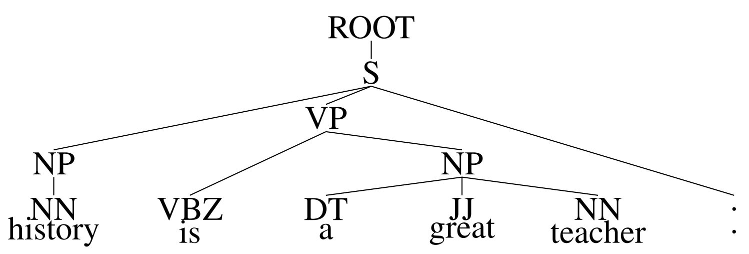


Figure 5.1: Example of a constituency parse tree before linearization.

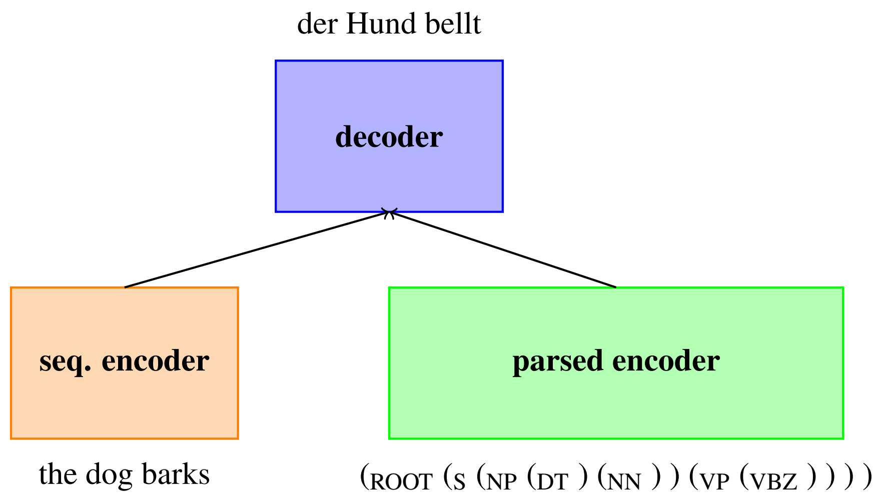


Figure 5.2: Illustration of the proposed multi-source model. Two encoders, the sequential (*seq.*) encoder and the parsed encoder, are used. Here, we show the unlexicalized version of the parse; the lexicalized version is similar.

5.3.2 Multi-Source Syntactic NMT

We propose using a multi-source framework (Zoph and Knight, 2016) to inject these linearized source parses into NMT. The proposed model consists of two identical LSTM encoders with no shared parameters, as well as a standard LSTM decoder. For each target sentence, two versions of the source sentence are used: the unparsed (standard) version and the linearized parse (lexicalized or unlexicalized). Each of the source sentence versions is encoded simultaneously using the encoders; the encodings are then combined and used as input to the decoder. Figure 5.2 shows an illustration of the multi-source model.

We combine the source encodings using the hierarchical attention combination proposed by Libovickỳ and Helcl (2017). At each timestep j in the decoder, a separate attention vector $\mathbf{c}_j$ is calculated over each encoder l independently as follows:

$$S_l(\mathbf{h}_i^{(l)}, \mathbf{s}_j) = \mathbf{v}_l^\top \tanh(\mathbf{W}_l \mathbf{h}_i^{(l)} + \mathbf{U}_l \mathbf{s}_j) \tag{5.1}$$

$$\alpha_j^{(l)}(i) = \frac{exp(S_l(\mathbf{h}_i^{(l)}, \mathbf{s}_j))}{\sum_k exp(S(\mathbf{h}_k^{(l)}, \mathbf{s}_j))} \tag{5.2}$$

$$\mathbf{c}_j^{(l)} = \sum_i \alpha_j^{(l)}(i)\mathbf{h}_i^{(l)} \tag{5.3}$$

where $\mathbf{s}_j$ hidden state of decoder and $\mathbf{v}_l$, $\mathbf{W}_l$ $\mathbf{U}_l$ are hyperparameters for each encoder. The two attention vectors are then projected into a shared vector space:

$$\mathbf{e}_j^{(l)} = \mathbf{v}_a^\top \tanh(\mathbf{W}_a\mathbf{s}_j + \mathbf{U}_a^{(l)}\mathbf{c}_j^{(l)}) \tag{5.4}$$

where $\mathbf{v}_a$ and $\mathbf{W}_a$ are shared hyperparameters and $\mathbf{U}_a^{(l)}$ are encoder-specific hyperparameters. Finally, the final attention vector $\mathbf{d}_j$ is calculated by attending to the two projected vectors:

$$\beta_j(l) = \frac{exp(\mathbf{e}_j^{(l)})}{\sum_k exp(\mathbf{e}_j^{(k)})} \tag{5.5}$$

$$\mathbf{d}_j = \sum_k \beta_j(k)\mathbf{U}_b^{(k)}\mathbf{c}_j^{(k)} \tag{5.6}$$

where $\mathbf{U}_b^{(k)}$ are encoder-specific hyperparameters. This multi-source method is thus able to combine the advantages of both syntactic encodings and standard sequential encodings.

5.4 Experimental Setup

5.4.1 Data

We run our experiments on the WMT17 (Bojar et al., 2017) English (EN)→German (DE) news translation task. All 5.9 million parallel training sentences are used, but no monolingual data is used for training. Validation is done on newstest2015, while newstest2016 and newstest2017 are used as test and held-out data, respectively.

We train a shared BPE (Sennrich et al., 2016d) vocabulary with 60k merge operations on the parallel training data. For the parsed data, we break words into subwords after applying the Stanford parser (Manning et al., 2014). We tokenize and truecase the data using the Moses tokenizer and truecaser (Koehn et al., 2007).

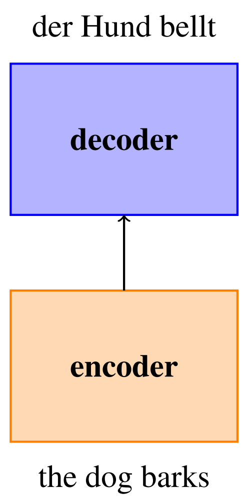


Figure 5.3: Seq2seq baseline model with a standard encoder and decoder.

5.4.2 Implementation and Training

The models are implemented in Neural Monkey (Helcl and Libovickỳ, 2017). They are trained using Adam (Kingma and Ba, 2015) and have minibatch size 40, RNN size 512, and dropout (Gal and Ghahramani, 2016) probability 0.2. We train to convergence on the validation set, using BLEU (Papineni et al., 2002) as the metric.

For unparsed inputs, the maximum sentence length is 50 subwords. For parsed inputs, we increase maximum sentence length to 150 subwords to account for the increased length due to the parsing labels.

5.4.3 Baselines

Seq2seq

The proposed models are compared against three baselines. The first, referred to here as *seq2seq*, is the standard LSTM-based neural machine translation system with attention (Bahdanau et al., 2015). This baseline is purely sequential; it does not use any parsed data. We show an example of this model in Figure 5.3.

Parse2seq

The second baseline we consider is a slight modification of the mixed RNN model proposed by Li et al. (2017). This baseline uses an identical architecture to the seq2seq baseline (except for a longer maximum sentence length in the encoder to account for the parsing tags). Instead of using sequential data on the source side, however, the

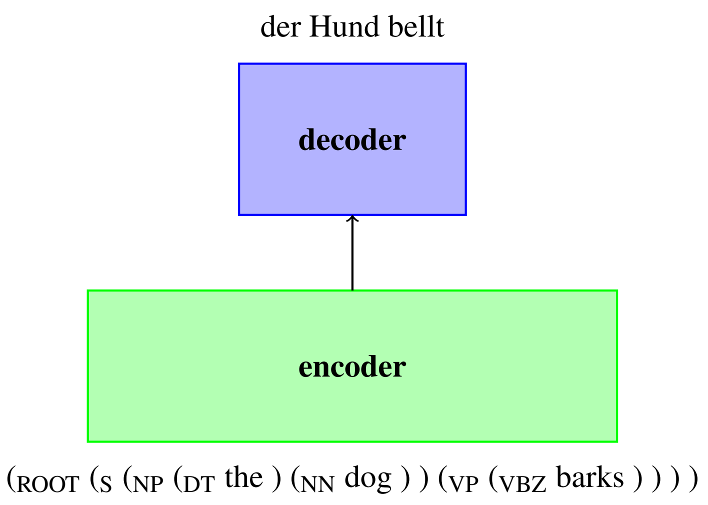


Figure 5.4: Parse2seq baseline model. A single encoder reads in parsed source sentences.

linearized, lexicalized parses are used. We allow the system to attend equally to words and node labels on the source side, rather than restricting attention to words. We refer to this baseline as *parse2seq*; it is shown in Figure 5.4.

Multi-Source Baseline

Finally, we compare our models to a multi-source unparsed baseline. This baseline consists of a multi-source system like the one described in section 5.3.2. However, both encoders encode the unparsed versions of the sentences (so both source sentences are identical). Therefore, this data does not use any parsed data. We refer to this baseline as *multi-source baseline*. Figure 5.5 depicts this baseline model.

5.5 Results

5.5.1 Main Experiments

Table 5.2 shows the performance on EN→DE translation for each of the proposed systems and the baselines, as measured by BLEU score.

Both proposed multi-source systems improve over all baselines, with improvements of up to 1.5 BLEU over the seq2seq baseline, up to 1.1 BLEU over the parse2seq baseline, and up to 0.8 BLEU over the multi-source baseline. In addition, the lexicalized multi-source system yields slightly higher BLEU scores than the unlexicalized

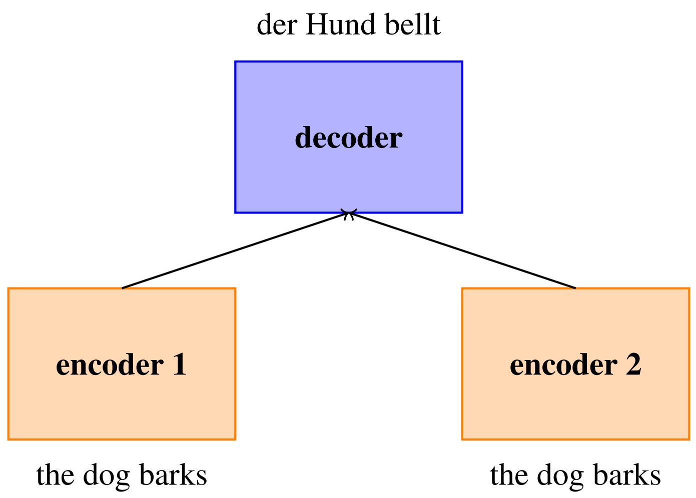


Figure 5.5: Multi-source baseline model. Both encoders encode the same unparsed source sentence simultaneously.

	System	**2016**	**2017**
baseline	seq2seq	25.0	20.8
	parse2seq	25.4	20.9
	multi-source baseline	25.7	21.4
proposed	lexicalized multi-source	**26.5**	**21.9**
	unlexicalized multi-source	26.4	21.7

Table 5.2: BLEU scores on the newstest2016 and newstest2017 datasets for the baselines and proposed unlexicalized and lexicalized multi-source systems.

multi-source system; this is surprising because the lexicalized system has significantly longer sequences than the unlexicalized one.

In summary, we can gain reasonable improvements in translation performance by allowing a model to see both parsed and sequential versions of the source sentence. It is also interesting to compare the seq2seq and parse2seq baselines. Parse2seq outperforms seq2seq by only a small amount compared to our multi-source models; thus, while adding syntax to NMT can be helpful, some ways of doing so are more effective than others. The multi-source baseline also improves over the standard seq2seq baseline despite not using any additional information, which implies that some of the improvements may be due to the increased number of parameters from having two encoders. However, the multi-source baseline still does worse than both proposed multi-source systems, indicating that they are an effective way of adding syntactic data to NMT.

5.5.2 Analysis

BLEU by Sentence Length

For models that rely on source-side linearized parses (multi-source and parse2seq), the source sequences are significantly longer than for the seq2seq baseline. Since NMT already performs relatively poorly on long sentences (Bahdanau et al., 2015), adding linearized source parses may exacerbate this issue. To detect whether this occurs, we calculate BLEU by sentence length.

We bucket the sentences in the held-out set (newstest2017) by source sentence length. We then compute BLEU scores for each bucket for the baselines and the lexicalized multi-source system. The results are in Figure 5.6.

In line with previous work on NMT on long sentences (Bahdanau et al., 2015; Li et al., 2017), we see a large deterioration in BLEU for all systems as sentence length increases. In particular, the parse2seq model, which outperformed the seq2seq model overall, does worse than seq2seq for sentences containing more than 30 words, indicating that parse2seq performance does indeed suffer due to its longer parsed sentences.

On the other hand, the lexicalized multi-source system outperforms the seq2seq and multi-source baselines for all sentence lengths and does particularly well compared to the baselines for sentences with over 50 words. This may be because the multi-source system uses both unparsed and parsed inputs, so it can rely more on unparsed inputs as source sentence length increases while still benefiting from the syntactic information

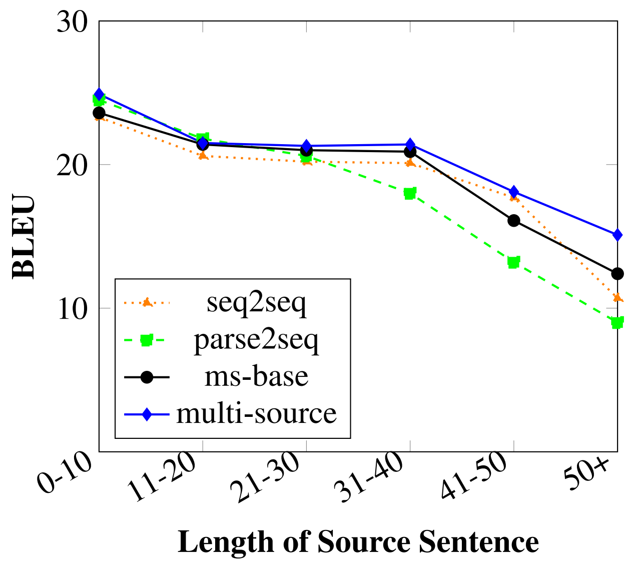


Figure 5.6: BLEU by sentence length on newstest2017 for baselines and lexicalized multi-source. *ms-base* refers to the multi-source baseline.

learned from the parsed inputs.

Inference Without Parsed Sentences

The parse2seq and multi-source systems require parsed source data at inference time. Here, we examine BLEU scores for these systems when they are given only unparsed source sentences at test time. This may be important when there are strict latency requirements during inference (leaving not enough time to parse source sentences) or when the parser fails at test time (e.g. when an out-of-domain sentence is used).

Table 5.3 displays the results of these experiments. For the parse2seq baseline, we use only unparsed data as input. For the lexicalized and unlexicalized multi-source systems, two options are considered: *unparsed* + *unparsed* uses identical unparsed data as input to both encoders, while *unparsed* + *null* uses null input for the parsed encoder, where every source sentence is simply *( )*.

The parse2seq system fails to produce any reasonable output when given only standard (unparsed) source data. On the other hand, both multi-source systems perform relatively well without parsed data, although the BLEU scores are worse than multi-source with parsed data. This indicates that our proposed multi-source systems are viable for cases when the test data cannot be parsed.

System	Test Source	2016	2017
parse2seq	unparsed	0.6	0.5
lexicalized multi-source	unparsed + unparsed	23.6	20.0
	unparsed + null	23.1	19.3
unlexicalized multi-source	unparsed + unparsed	**23.7**	19.9
	unparsed + null	23.6	**20.9**

Table 5.3: BLEU scores on newstest2016 and newstest2017 when no parsed data is used during inference.

System	Test Source	2016
multi-source lex	parsed	26.8
	unparsed	26.7
multi-source unlex	parsed	**26.9**
	unparsed	26.8

Table 5.4: BLEU scores on newstest2016 when half of the training data is left unparsed.

Training Without Parsed Sentences

In the multi-source experiments described above, we used parsed and unparsed versions of each training sentence and discarded the training sentence for which the parser failed. In this section, we examine how robust the multi-source experiments are to parser failure during training.

For each parsed source sentence in the training data, we replace the parsed version with the null version *( )* with a probability of 0.5, meaning that roughly 50% of the source training data seen by the parsed encoder is actually blank. We do not make any modifications to the unparsed data, so the unparsed encoder sees an unparsed source version for each sentence. We then train the multi-source systems as before.

We evaluate each model in two ways. The first, *parsed*, uses only parsed source sentences as input to the parsed encoder; the second, *unparsed*, uses only null inputs to the parsed encoder. In both cases, the unparsed encoder uses unparsed inputs. The results for these experiments are shown in Table 5.4.

When roughly half of the sentences are unparsed during training, the models yield as good of translation performance as the original multi-source models (see Table 5.2) and continue to outperform the baselines. In addition, Table 5.4 shows that these models do just as well when not using parsed source sentences for inference as when using

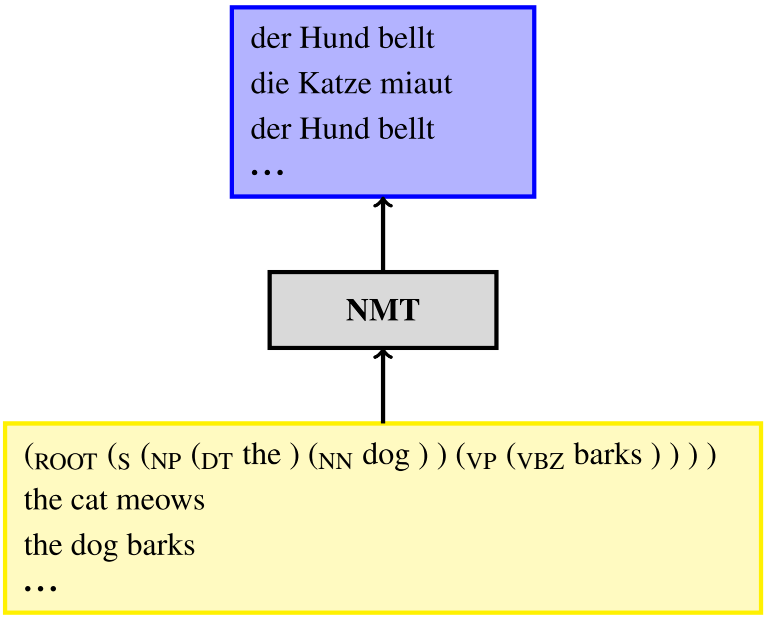


Figure 5.7: Illustration of the proposed shared encoder model. Translation is done from either the parsed or the unparsed source sentence.

them. This indicates that it would be possible to add unparsed data (e.g. synthetic source sentences or the copied corpus described in chapter 3) during training using blank parsed sentences.

5.5.3 Extension: Shared Encoder Model

Inspired by the results in the previous section for training without parsed sentences, we propose an extension to the multi-source model, the *shared encoder* model. In order to train this model, we create two copies of the training data, one with lexicalized parsed source sentences and the other with unparsed source sentences. We then shuffle these training corpora together and train the system like normal, with a shared encoder for both parsed and unparsed source sentences. Thus, the model is trained to translate both from parsed and from unparsed sources. Figure 5.7 illustrates this shared encoder model.

The shared encoder model is similar to the proposed multi-source models. The multi-source models generate each translation from the parsed and unparsed versions of the same source sentence; this shared encoder model generates translations from either the parsed version or the unparsed version of the source sentence. Unlike the multi-source models, the shared encoder model does not use unlexicalized parses, since it is unlikely that the model could learn to translate a sentence without information

System	Test Source	2016
multi-source baseline	unparsed + unparsed	25.7
lexicalized multi-source	unparsed + parsed	26.5
shared encoder	parsed	26.6
	unparsed	**27.5**

Table 5.5: Results for the shared encoder model with parsed and unparsed source sentences. We include the results from the best baseline (the multi-source baseline) and the best multi-source model (lexicalized multi-source) for comparison.

about the source words.

The motivation for this model is twofold. First, we introduce this model by analogy with the many-to-one multilingual translation experiments of Johnson et al. (2017). Instead of training the system to translate from two different source languages, we train it to translate from two different source formats (parsed and unparsed). In addition, since the model is trained explicitly to translate both parsed and unparsed inputs, we expect that it will be able to use both parsed and unparsed data at inference time. Thus, the model can take advantage of the parses to gain information about the source language structure, while still not relying on the parses at inference time.

Table 5.5 shows the results for the shared encoder model using both parsed and unparsed sentences as source sentences during inference. We also display the multi-source baseline and lexicalized multi-source results for comparison.

The shared encoder model achieves even higher BLEU scores (by 1.0 BLEU) than our proposed multi-source models when no parsed data is used during inference. When parsed source data is used during inference, the results are comparable to the multi-source models. In all cases, the shared encoder model outperforms the baselines. The fact that inference with unparsed inputs does better than inference with parsed inputs means that we do not have to worry about increased latency or parser availability at inference time. In the next chapter, we use the shared encoder model as a baseline for our multi-task syntactic NMT system and apply it to a new architecture, the transformer.

5.6 Discussion

Our multi-source syntactic NMT models achieved higher BLEU scores than both syntactic and non-syntactic baselines and were better able to translate long sentences than the baselines. We attribute these improvements to the fact that the multi-source models have the best of both worlds. The syntactic annotations can help them learn the structure of the source sentence, but the models can also take advantage of the shorter unparsed source sentences when necessary. In addition, our multi-source models were able to translate unparsed inputs reasonably well because they had seen both unparsed and parsed sentences during training.

Based on our results, we proposed two extensions to the multi-source model that yielded further improvements. First, we trained the multi-source model with some empty parses, so that the model learned to translate both from the parsed + unparsed sentence combination and from the unparsed sentences only. This allowed it to improve inference on unparsed sentences while still preserving the gains in BLEU over the baselines. We also introduced a shared encoder model that used a single encoder for both translation from parsed sentences and translation from unparsed sentences. This model showed even better translation performance than the multi-source syntactic models.

The main limitation of the work in this chapter is the dependence on an external parser. Although we showed ways to train the model to be robust to parser failure, we were still required to parse millions of parallel training sentences. The models proposed here would not be feasible for translating from lower-resource languages that do not have parsers. In addition, although we showed strong improvements over relevant baselines for our multi-source and shared encoder systems, our models could still be further improved by using monolingual data and ensembles of multiple systems.

5.7 Conclusions

This chapter proposed a novel strategy for improving neural machine translation using linearized source parses. Our multi-source method, in which parsed and unparsed versions of the same source sentence were embedded using separate encoders, resulted in gains of up to 1.5 BLEU on EN→DE translation. In addition, it was able to translate reasonably well even when the source sentence was not parsed, which is useful in the case that the external parser has failed on a test sentence. It also showed strong performance on long sentences relative to the baselines.

Based on the success of the multi-source method in dealing with both parsed and unparsed inputs, we proposed a second method, which we called *shared encoder*. This method was trained on a mix of parsed and unparsed source sentences, as well as unparsed target sentences. The shared encoder method yielded even higher BLEU scores than the multi-source method.

The goal of this thesis is to propose ways of using non-parallel training data to improve NMT. In the previous two chapters, we explored using monolingual corpora as our additional non-parallel training data. Here, we study a new tactic: using an external constituency parser as another source of information. Unlike for the target monolingual data case (chapter 3), we have seen here that adding syntactic information can help in high-resource scenarios. In subsequent chapters, we will turn our attention to adding syntax in low-resource scenarios and cases where a parser might be unavailable for the source language.

In the future, we would like to further study our multi-source and shared encoder models with respect to their syntactic behavior. Even though both models yielded stronger BLEU scores than the baselines, we do not yet have a good understanding of how the additional syntactic information helps. For example, it could be illuminating to look at attention weights for different sentences or words in the multi-source model to see in what cases the model attends to parsed vs. unparsed source sentences. In addition, we could see how well the models can predict syntactic or structural information on the target side. In chapter 6, we introduce a multi-task syntactic NMT model and analyze whether it learns to generate parses with balanced parentheses (section 6.6); similar analysis would be interesting here.

Although our techniques showed improvements over seq2seq, parse2seq, and multi-source baselines, they did not improve over the state of the art because we used only parallel data to train all the systems in this chapter. An important avenue for future work will be exploring ways of adding monolingual data to the multi-source systems; this could take the form of back-translations (Sennrich et al., 2016c) or copied monolingual data (chapter 3). Adding target monolingual data is a challenge for NMT methods that use source syntax, because the external parser might be unreliable for the synthetic source sentences. However, the models presented in this chapter do not require all training data to be parsed, making them a promising method for this task.

Chapter 6

Incorporating Source Syntax into Transformer-Based NMT

This chapter builds on the previous chapter by examining whether source-side syntax can also be helpful in a transformer-based neural machine translation setting. We propose a multi-task model for incorporating syntax into the transformer, which continues to use the linearized parses introduced in chapter 5. This chapter is based on Currey and Heafield (2019).

6.1 Introduction

In this chapter, we expand our methods for linearized parse-based syntactic neural machine translation to a transformer architecture. There are two main motivations for doing this. First, transformers have shown improvements over RNN-based NMT models in many tasks (Bojar et al., 2018); as the field advances, we would like to test whether syntactic NMT with linearized parses is still helpful on new architectures. Second, transformer-based NMT may stand to benefit as much from explicit syntactic annotations as RNN-based NMT, particularly in low-resource settings. On the one hand, the transformer model already learns some syntax without explicit supervision in high-resource cases. Vaswani et al. (2017) visualized a few encoder self-attentions in a trained NMT model and found that they seemed to capture syntactic structure. This was formalized by Raganato and Tiedemann (2018), who found that transformer encoders trained on high-resource NMT were able to perform reasonably well at part-of-speech tagging, chunking, and other tasks. However, for transformers trained on low-resource NMT, the results on these tasks were not as strong. Additionally, Tran et al. (2018) found that a transformer language model did not do better at predicting subject-verb agreement than an RNN language model; Tang et al. (2018) saw similar results for transformer vs. RNN NMT models.

Thus, the goal of this chapter is to improve transformer-based NMT using source-side syntactic supervision. We investigate two methods that incorporate source-side linearized constituency parses into transformer-based NMT. The first, shared encoder, was introduced in section 5.5.3; it learns to translate directly from both parsed and unparsed source sentences. Since the shared encoder method performed better than the multi-source model introduced in chapter 5, we use it as a baseline in this chapter and test its applicability to a new architecture (the transformer). We also propose a second method, multi-task, that uses the transformer to learn to parse and translate the source sentence simultaneously. We empirically evaluate both methods on translation from English into 21 diverse target languages, finding that the multi-task method improves consistently over a non-syntactic baseline for low-resource NMT.

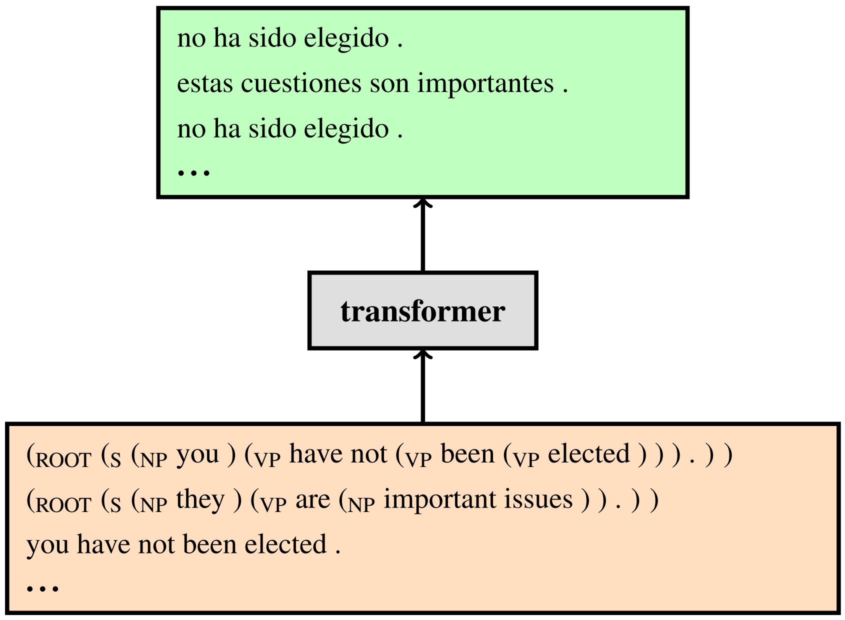
6.2 Transformer-Based NMT with Linearized Parses

We propose a multi-task model for incorporating linearized parses into transformer-based NMT and further investigate the shared encoder model that was briefly introduced in section 5.5.3. Both methods rely on linearized parses of the source sentences to inject source syntax into transformer-based NMT. We described the process for linearizing parses in section 5.3.1; we use the same process for these methods, except that here we remove part-of-speech tags from the parses in order to shorten the lengths of the parsed sequences (as was done by Aharoni and Goldberg, 2017). (We ran initial experiments for the multi-task models with part-of-speech tags, but found that their performance was slightly worse than the models without the tags.)

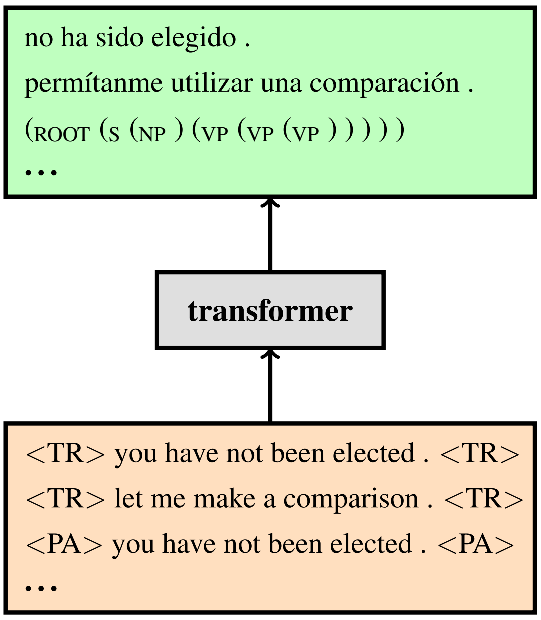
Figure 6.1 summarizes the two proposed methods. They are discussed in detail in sections 6.2.1 and 6.2.2, respectively.

6.2.1 Shared Encoder Transformer

We start by reviewing the shared encoder model introduced in section 5.5.3. Since this method was successful at incorporating linearized source parses into RNN-based NMT, we use it as a baseline here and study its applicability to the transformer architecture. The shared encoder model learns to translate from both unparsed and parsed


no ha sido elegido .
estas cuestiones son importantes .
no ha sido elegido .
...
transformer
(ROOT (S (NP you) (VP have not (VP been (VP elected))) .))
(ROOT (S (NP they) (VP are (NP important issues)) .))
you have not been elected .
...

(a) Shared encoder syntactic NMT model. The system learns to translate directly from both parsed and unparsed source sentences into unparsed target sentences.


no ha sido elegido .
permítanme utilizar una comparación .
(ROOT (S (NP) (VP (VP (VP)))))
...
transformer
<TR> you have not been elected . <TR>
<TR> let me make a comparison . <TR>
<PA> you have not been elected . <PA>
...

(b) Multi-task syntactic NMT model. The system is trained to translate (<TR>) and parse (<PA>) source sentences using the same architecture.

Figure 6.1: Illustrations of the two proposed syntactic NMT methods.

($_{\text{ROOT}}$ ($_{\text{S}}$ ($_{\text{NP}}$ you) ($_{\text{VP}}$ have not ($_{\text{VP}}$ been ($_{\text{VP}}$ elected))) .))	$\rightarrow$ no ha sido elegido .
you have not been elected .	$\rightarrow$ no ha sido elegido .

Table 6.1: Example of the two formats of English$\rightarrow$Spanish training data for the shared encoder system.

source sentences into unparsed target sentences.

In order to train the shared encoder model, we create two copies of the training data, one with parsed source sentences and the other with unparsed source sentences. We then shuffle these training corpora together into a single corpus and train a standard transformer NMT model on the final data, with a single shared encoder for both parsed and unparsed source sentences. The training data contains (parsed source, unparsed target) and (unparsed source, unparsed target) sentence pairs; Table 6.1 gives an example of the two types of training sentence pairs for the shared encoder method. Since the data is shuffled, these two sentence pairs (with identical target sentences) will not necessarily be seen together during training.

Since the model is trained on both parsed and unparsed source sentences, during inference it is able to translate from either source sentence format. Inference on unparsed source sentences is slightly faster (since it does not require parsing of the source sentence) and achieves higher BLEU scores (as seen in Table 5.5), so we show results using unparsed source sentences for our experiments (sections 6.4.2 and 6.5.2).

6.2.2 Multi-Task NMT and Parsing with Shared Decoder

Our main proposal in this chapter adopts a multi-task framework to incorporate source-side syntax into transformer-based NMT. The primary task is translating the source sentence into the target language; the secondary task is parsing the source sentence using the seq2seq parsing technique of Vinyals et al. (2015). For the parsing task, we employ the same encoder-decoder framework as for NMT, with the sequential source sentence as input and the linearized, unlexicalized parsed source sentence as output. Thus, both tasks are trained using a single model with a shared encoder and decoder. This is similar to the multi-task framework proposed by Luong et al. (2016), with three main differences: 1) we do not use separate decoders for each task, 2) we use the same source data for both parsing and translation, and 3) we use a transformer rather than recurrent neural network architecture.

We do not directly use gold parses to train the parsing task, nor do we split the

translation
<TR> you have not been elected . <TR> → no ha sido elegido .

parsing
<PA> you have not been elected . <PA> → (ROOT (S (NP) (VP (VP (VP)))))

Table 6.2: Example of English→Spanish training data for translation and parsing tasks in the multi-task system.

training data between the two tasks. The reason for using the parallel corpus as the basis for the training data for both tasks is that we expect it to be difficult to find a sufficiently large amount of in-domain gold parses for training; additionally, our main goal is to improve NMT, so we do not expect the lower quality of the synthetic parses to matter.

In order to generate the training data for this model, we first create linearized parses of the source side of the training corpus. Next, we add a tag at the beginning and end of each source sentence indicating the desired task, similar to what was done by Johnson et al. (2017) for multilingual NMT. Table 6.2 gives an example of the data format. Finally, we shuffle the parsing and translation training data together and train the shared encoder and decoder on both tasks, making no further distinction between the tasks during training. Since we parse all of the training data, each source sentence appears twice: once with a target-language sentence and once with a parse of the source sentence. These copies are shuffled separately.

6.3 Experimental Setup

6.3.1 Data

We preprocess our data for all experiments as follows. First, we tokenize and truecase the data using the Moses scripts (Koehn et al., 2007). We then train BPE (Sennrich et al., 2016d) vocabularies with 30k merge operations. We use the Stanford CoreNLP parser (Manning et al., 2014) to generate constituency parses of the source (English) sentences; we linearize and format the parses as described in section 5.3.1 and additionally remove the part-of-speech tags. To create the shared encoder training data, we apply BPE to the parses and mix the parsed source data with the unparsed source data. For the multi-task training data, we remove words from the parses and shuffle the (source, target) and (source, parse) training corpora together. We do not use monolingual training data. For the multi-task experiments, validation is done on the translation task only (not on parsing), and for the shared encoder experiments, validation is done on unparsed source sentences only. Sections 6.4.1 and 6.5.1 contain detailed information on the target languages and data used.

6.3.2 Implementation and Training

All models are implemented in Sockeye (Hieber et al., 2017). For hyperparameter settings, we follow the recommendations of Vaswani et al. (2017). For unparsed inputs and outputs, sentence length is capped at 50 subwords, except in the case of the multi-task systems, where sentences of up to 52 subwords are allowed because of the parsing/translation tags. As in section 5.4.2, we allow a maximum length of 150 subwords for parsed sentences. We evaluate our multi-task and shared encoder models compared to a standard (non-syntactic) transformer baseline on translation from English into the 21 target languages.

6.4 Small-Scale Cross-Lingual Experiments

6.4.1 Data

We use the Europarl Parallel Corpus (Koehn, 2005) as the basis for our small-scale cross-lingual experiments. We consider translation from English (EN) into each of the twenty remaining target languages; Table 6.3 contains a full list of the target languages, as well as their language families or branches. For these experiments, we train separate subword vocabularies for the source and for each target language, in order to allow the source data to be identical for all experiments. By using this data set, we are able to evaluate the usefulness of syntactic information for several relatively diverse target languages. However, all the languages in these experiments are Indo-European or Uralic, since we are using Europarl.

In order to facilitate comparison between the target languages, we follow Cotterell et al. (2018) by taking only the intersections of the Europarl training data. This means that the source (EN) data is identical for all experiments, and the targets are all translations of each other in the different target languages. This results in 170k parallel training sentences for each language pair. We reserve a random subset of 10k sentences

Family	**Language**	**Abbreviation**
Baltic	Latvian	LV
	Lithuanian	LT
Germanic	Danish	DA
	Dutch	NL
	German	DE
	Swedish	SV
Hellenic	Greek	EL
Romance	French	FR
	Italian	IT
	Portuguese	PT
	Romanian	RO
	Spanish	ES
Slavic	Bulgarian	BG
	Czech	CS
	Polish	PL
	Slovak	SK
	Slovene	SL
Uralic	Estonian	ET
	Finnish	FI
	Hungarian	HU

Table 6.3: Target languages used in our experiments, along with their language families or branches and their abbreviations.

from the original data to use as development data and an additional 10k sentences as test data.

6.4.2 Results

Table 6.4 displays BLEU scores on the test data for each target language for the proposed systems. The multi-task system outperforms the non-syntactic baseline for all target languages. In addition, for all but four target languages (SV, EL, SK, and ET), the multi-task system is at least 1 BLEU point better than the baseline. Thus, our proposed multi-task method consistently improves over a non-syntactic baseline across several diverse target languages in low-resource scenarios. Additionally, in all cases but one (EN→ET), the multi-task model achieves the highest BLEU score of all models.

The performance of the shared encoder system in relation to the non-syntactic baseline is less consistent than that of the multi-task systems. In most cases, shared encoder improves only slightly (less than 1 BLEU) over the baseline, although for LV, LT, RO, ES, PL, and FI, the improvements are stronger. However, for four target languages (NL, EL, BG, and SK), the shared encoder system does worse than the non-syntactic baseline.

Target language family does not seem to have a noticeable effect on the performance of either the shared encoder or the multi-task method; this could be due to the fact that the syntactic annotations were on the source sentence only. It remains to be seen whether certain source languages are particularly amenable to incorporating source syntax in NMT.

6.5 Full-Scale WMT Experiments

6.5.1 Data

The main goal of the previous section was to evaluate our proposed syntactic NMT methods on a wide range of target languages and see whether target language or language family had an effect performance. In this section, we run additional experiments in order to evaluate the proposed methods on a standard set of benchmarks. We train our models on the following tasks: English→Turkish (TR) from the WMT18 news translation shared task (Bojar et al., 2018), English→Romanian WMT16 (Bojar et al., 2016), and English→German WMT17 (Bojar et al., 2017).

EN→*	**base**	**shared encoder**	**multi-task**
LV	26.5	28.1 (+1.6)	**28.2** (+1.7)
LT	23.5	24.6 (+1.1)	**24.8** (+1.3)
DA	39.5	40.1 (+0.6)	**40.7** (+1.2)
NL	28.8	28.7 (-0.1)	**30.6** (+1.8)
DE	30.5	30.6 (+0.1)	**32.1** (+1.6)
SV	35.9	**36.4** (+0.5)	**36.4** (+0.5)
EL	38.9	38.8 (-0.1)	**39.7** (+0.8)
FR	38.3	38.5 (+0.2)	**40.4** (+2.1)
IT	31.3	31.3 (==)	**32.5** (+1.2)
PT	39.2	39.3 (+0.1)	**40.5** (+1.3)
RO	36.3	**37.8** (+1.5)	**37.8** (+1.5)
ES	41.6	43.0 (+1.4)	**43.1** (+1.5)
BG	39.0	38.6 (-0.4)	**40.5** (+1.5)
CS	27.5	28.3 (+0.8)	**28.8** (+1.3)
PL	23.7	24.8 (+1.1)	**25.1** (+1.4)
SK	32.8	32.5 (-0.3)	**32.9** (+0.1)
SL	33.3	34.2 (+0.9)	**34.9** (+1.6)
ET	20.2	**20.9** (+0.7)	20.8 (+0.6)
FI	21.5	22.8 (+1.3)	**23.3** (+1.8)
HU	22.3	22.6 (+0.3)	**23.4** (+1.1)

Table 6.4: BLEU scores on the test set for small-scale cross-lingual experiments. The numbers in parentheses show difference with the non-syntactic baseline. All systems use EN as the source language.

System	newstest2017	newstest2018
baseline	9.6	8.8
shared encoder	9.6 (==)	9.3 (+0.5)
multi-task	**10.6** (+1.0)	**10.4** (+1.6)

Table 6.5: BLEU scores (and improvement over the baseline) for EN→TR on the test (newstest2017) and held-out (newstest2018) datasets.

For each experiment, we use all available parallel training data from the task, but no monolingual training data. This gives us 200k parallel training sentences for EN→TR, 600k for EN→RO, and 5.9M for EN→DE. Note that the EN→RO and EN→DE training corpora contain some overlaps with the Europarl training data from section 6.4.1, although the experiments in this section use significantly more training data. We validate EN→TR on newstest2016, EN→RO on newsdev2016, and EN→DE on newstest2015.

6.5.2 Results

The results for the EN→TR experiments are displayed in Table 6.5. These results mirror what was seen in the previous experiments: the shared encoder method gives modest improvements over the non-syntactic baseline (0.0–0.5 BLEU), while the multi-task method yields the strongest results, with an improvement of 1.0–1.6 BLEU points over the baseline. Although Turkish is not related to any of the target languages studied in section 6.4, the amount of training data for EN→TR is similar to what was used in that section, which might be one explanation for the similar results.

Table 6.6 shows the performance of each model on the WMT EN→RO experiments. Here, we see more modest improvements from adding the syntactic data: only 0.5 BLEU over the non-syntactic baseline for both the shared encoder and multi-task methods. It is interesting to compare this with the results for the Europarl EN→RO experiments (section 6.4.2); there, we saw a much larger improvement over the baseline for the multi-task model (1.5 BLEU). This indicates that the effectiveness of the multi-task model may depend on amount of data (the WMT models were trained on about three times as much training data) rather than on target language.

Finally, we display our WMT EN→DE results in Table 6.7. In this case, very high-resource EN→DE translation, the multi-task method does much worse than the non-syntactic baseline (-2.1 BLEU points). In addition, the shared encoder method achieves

System	**newstest2016**
baseline	21.5
shared encoder	**22.0** (+0.5)
multi-task	**22.0** (+0.5)

Table 6.6: BLEU scores (and improvement over the baseline) for EN→RO on the test set (newstest2016).

System	**newstest2016**	**newstest2017**
baseline	31.7	25.5
shared encoder	**31.9** (+0.2)	**26.0** (+0.5)
multi-task	29.6 (-2.1)	23.4 (-2.1)

Table 6.7: BLEU scores (and difference with the baseline) for EN→DE on the test (newstest2016) and held-out (newstest2017) datasets.

comparable BLEU scores to the baseline (only 0.2–0.5 BLEU higher). Thus, neither proposed technique is particularly successful for high-resource EN→DE NMT. Again, we can contrast this with the Europarl EN→DE experiments, where we saw strong improvements from the multi-task model (1.6 BLEU). This lends further credence to the hypothesis that these NMT models with linearized source parses are helpful cross-linguistically in low-resource scenarios, but not in high-resource setups.

It is interesting to compare these EN→DE results to the results in Table 5.5. There, we also used a shared encoder model, but with an RNN architecture rather than a transformer. The RNN-based shared encoder model outperformed the non-syntactic multi-source RNN baseline by 1.1-1.8 BLEU, much more than is the case for the transformer-based models. However, all of the RNN-based models had much lower BLEU scores than the transformer models.

We further investigated the WMT EN→DE multi-task model to find reasons for the large drop in performance compared to the non-syntactic baseline. We found that while the multi-task model was able to generate reasonable (albeit lower-quality) translations, it did not successfully learn to parse. During parsing inference, the model learned to always output the same parse regardless of the input sentence: *(ROOT (S (NP) (VP (NP (NP) (PP (NP (NP) (PP (NP)))))))))* . This was a common parse in the training data (it occurred 12k times in the data). This issue is partially due to the fact that validation is only done on the translation task, not on the parsing task. However, we do not see this issue with the other language pairs and experiments. This failure to learn to parse indicates that the WMT EN→DE multi-task model is not able to take advantage of the syntactic annotations.

6.6 Analysis: Validity of Parses

The multi-task syntactic NMT models are trained both to translate and to parse the input sentences. The main goal of these models has been to improve translation; those results were reported in sections 6.4.2 and 6.5.2. In this section, we analyze the validity of the parses produced by the multi-task systems. We use a standard parsing benchmark, Wall Street Journal section 23 of the Penn Treebank (Marcus et al., 1993), as the evaluation dataset in this section. We preprocess this dataset as described in section 6.3 before inputting it into the multi-task system.

The multi-task models were trained to generate unlexicalized parses. Since we removed part-of-speech tags from the parses during preprocessing, it is not possible to automatically relexicalize the parses. This is because there is no one-to-one correspondence between the number of leaves of the parse tree and the number of words in the sentence. Thus, rather than evaluating the parses directly, we count the number of valid parses (i.e. parses with balanced parentheses) per target language.

Table 6.8 shows the percent of generated parses that are valid for the Europarl multi-task models. For most target languages, over 90% of the generated parses are valid.

Unlike for the translation results, target language family does seem to have an effect on the parsing results. Overall, Romance, Germanic, and Hellenic target language systems generate the fewest valid parses. This indicates that Baltic, Slavic, and Uralic target languages are most helpful in learning to parse English in a multi-task system. Thus, from our cross-lingual experiments, it seems that the parsing performance of a multi-task system depends somewhat on target language family, whereas we saw in the previous sections that the translation success depends more on the amount of training data. Note, however, some caveats: 1) we did not perform validation on the parsing task (only on the translation task), and 2) we are measuring only parsing validity here, rather than parsing performance.

Table 6.9 shows the percent of valid parses for the three WMT multi-task experiments. For EN→DE, all of the generated parses are valid because they are all identical (as dicussed in section 6.5.2). For EN→RO, nearly all the parses are valid as well.

EN→*	**% Valid Parses**
LV	96.8%
LT	99.2%
DA	70.8%
NL	93.3%
DE	87.2%
SV	95.4%
EL	85.2%
FR	92.3%
IT	78.8%
PT	89.4%
RO	96.3%
ES	86.5%
BG	97.5%
CS	95.9%
PL	98.1%
SK	98.5%
SL	97.3%
ET	98.2%
FI	95.1%
HU	93.6%

Table 6.8: Percent of valid parses of the parses generated by the Europarl multi-task systems.

EN→*	**% Valid Parses**
TR	86.3%
RO	99.8%
DE	100%

Table 6.9: Percent of valid parses of the parses generated by the WMT multi-task systems.

However, this language pair did not have the same issue as EN→DE: the parses generated for each sentence were different, and a manual analysis indicated that the generated EN→RO parses were reasonable. The EN→TR system generated a large amount of valid parses, but fewer than the EN→RO system; it is possible that the EN→TR system would have done better with more training data.

6.7 Discussion

Our multi-task model showed strong improvements across twenty target languages when trained on a small-scale task; however, as more data was added, the multi-task model did worse in relation to the baseline. The shared encoder model was more inconsistent on the small-scale tasks but did slightly better than the baselines on the larger-scale WMT EN→DE task. This is an interesting result because the shared encoder model did much better than an RNN baseline for the same EN→DE task in chapter 5; it seems that this shared encoder model had a harder time beating the much stronger transformer baseline here.

The two methods studied in this chapter are straightforward to apply to different architectures, since they rely only on linearized versions of syntactic parses. Neither method requires data to be parsed at inference time, which can save translation time. Additionally, the multi-task model is helpful across several language pairs in low-resource NMT scenarios. On the other hand, neither model was particularly successful compared to non-syntactic NMT for high-resource scenarios. Our models also relied on an external parser to create parsed parallel training data; for a truly low-resource source language with no parser, we would not be able to apply these models.

6.8 Conclusions

In this chapter, we have studied two methods for incorporating source-side syntactic annotations into a transformer-based neural machine translation system. The first, multi-task, uses a single encoder and decoder to train two tasks: translation and constituency parsing. The second, shared encoder, learns to translate both linearized parses of the source sentences and unparsed source sentences directly into unparsed target sentences. We performed experiments from English into twenty target languages in a low-resource setup; the multi-task system improved over the non-syntactic baseline for all target languages. We further demonstrated the success of this method on the EN$\rightarrow$TR and EN$\rightarrow$RO WMT datasets; however, for the very high-resource EN$\rightarrow$DE WMT setup, the multi-task model performed poorly. The shared encoder model yielded some improvements in BLEU overall, but they tended to be small (less than 1.0 BLEU).

The goal of this thesis is to incorporate non-parallel data into NMT training. In the first half of the thesis, we focused on monolingual data; here, we add syntactic information. This builds on the work in the previous chapter by expanding syntactic NMT to transformer architectures and introducing a multi-task model that still uses linearized parses to represent source syntax. Linearized parses have the advantage that they can easily be applied to any encoder architecture. In the next chapter, we will consider injecting structure into NMT without any explicit syntactic annotation.

In the future, we plan on extending these techniques to incorporate target-side syntax into transformer-based NMT. In addition, we would like to experiment with different source languages in order to find out whether adding source-side syntax has a greater effect on some source languages than others. Finally, the multi-task method could easily be combined with back-translation (Sennrich et al., 2016c) and with the copied corpus method (chapter 3) without needing to parse the synthetic source sentences; we believe that combining our syntactic NMT methods with methods for incorporating monolingual data into NMT could lead to improvements in the state of the art.

Chapter 7

Unsupervised Hierarchical Encoder for Neural Machine Translation

In this chapter, we consider the goal of adding structure to the neural machine translation encoder when no syntactic parser is available. We propose an unsupervised hierarchical encoder based on the Gumbel tree-LSTM of Choi et al. (2018). This chapter is based on Currey and Heafield (2018b).

7.1 Introduction

In chapters 5 and 6, we introduced ways of incorporating syntactic information about the source language into neural machine translation. The proposed methods made use of linearized constituency parses during training; the constituency parses were generated by applying an off-the-shelf parser to the source side of the parallel corpora. We showed that the proposed models, multi-source (chapter 5), shared encoder (chapters 5 and 6), and multi-task (chapter 6), are able to leverage this syntactic information to improve overall translation performance.

However, in the previous two chapters, we only performed experiments using English as the source language. For translation out of English or a few other high-resource languages, these syntactic NMT methods are viable because high-quality constituency parsers exist for those languages. On the other hand, for low-resource languages, a parser might not be available, making it impossible to apply these and other syntatic NMT methods. This is particularly frustrating because neural machine translation already does relatively poorly in low-resource scenarios (Koehn and Knowles, 2017), and because low-resource cases are exactly the cases where NMT models have most difficulty learning syntax (Raganato and Tiedemann, 2018). Adding source-side syntactic information has the potential to improve low-resource NMT, but we would need a way of doing so without an external parser.

We would like to mimic the improvements that come from adding source syntactic structure to NMT without needing syntactic annotations of the training data. Recently, there have been some proposals to induce unsupervised hierarchies over sentences based on semantic objectives for sentiment analysis and natural language inference (Choi et al., 2018; Yogatama et al., 2017). Here, we apply these hierarchical sentence representations to low-resource neural machine translation.

In this work, we replace the standard NMT encoder with the Gumbel tree-LSTM proposed by Choi et al. (2018) and apply this model to low-resource translation. This allows unsupervised hierarchies to be injected into the encoder. In addition, the Gumbel tree-LSTM has the advantage that it is fully differentiable, so it can make discrete parsing decisions without requiring reinforcement learning or other changes to the training paradigm. We compare this model to sequential neural machine translation, as well as to NMT enriched with information from an external parser. Our proposed model yields strong improvements in very low-resource NMT without requiring parsers or outside data beyond what is used in standard NMT. This work is the first to apply the Gumbel tree-LSTM model to neural machine translation.

7.2 Unsupervised Tree-to-Sequence NMT

We modify the LSTM-based neural machine translation architecture by combining a sequential LSTM decoder with a Gumbel tree-LSTM (Choi et al., 2018) encoder. This encoder induces a binary tree structure on the source sentence without syntactic supervision. We refer to our proposed models containing this encoder as *(unsupervised) tree2seq*.

In this section, we present our unsupervised tree2seq model. Section 7.2.1 describes the subword-level representations, while section 7.2.2 explains how the Gumbel tree-LSTM is used to add hierarchies in the encoder. We address top-down representations of the word and phrase nodes in section 7.2.3 and explain the attention mechanism in section 7.2.4.

7.2.1 Word Node Representations

The hierarchical encoder that we propose consists of *word nodes* (nodes corresponding to the subwords of the source sentence) and *phrase nodes* (internal nodes resulting from the induced hierarchies). In order to obtain representations of the word nodes, we run a bidirectional LSTM over the source sentence. We refer to this biLSTM as the *leaf LSTM*.

7.2.2 Phrase Node Representations

Our proposed unsupervised hierarchical encoder uses a Gumbel tree-LSTM (Choi et al., 2018) to obtain the representations of the phrase nodes in the source sentence. This encoder leverages the straight-through Gumbel softmax estimator (Jang et al., 2017) to induce unsupervised, discrete hierarchies over the source sentence without modifying the maximum likelihood objective used to train NMT.

In a Gumbel tree-LSTM, the hidden state $\mathbf{h}_p$ and memory cell $\mathbf{c}_p$ for a given node are computed recursively based on the hidden states and memory cells of its left and right children ($\mathbf{h}_l$, $\mathbf{h}_r$, $\mathbf{c}_l$, and $\mathbf{c}_r$). This is done as in a standard binary tree-LSTM (Tai et al., 2015) as follows:

$$\begin{bmatrix}\mathbf{i}\\ \mathbf{f}_l\\ \mathbf{f}_r\\ \mathbf{o}\\ \mathbf{g}\end{bmatrix} = \begin{bmatrix}\sigma\\ \sigma\\ \sigma\\ \sigma\\ \tanh\end{bmatrix}\left(\mathbf{W}\begin{bmatrix}\mathbf{h}_l\\ \mathbf{h}_r\end{bmatrix} + \mathbf{b}\right) \tag{7.1}$$

$$\mathbf{c}_p = \mathbf{f}_l \odot \mathbf{c}_l + \mathbf{f}_r \odot \mathbf{c}_r + \mathbf{i} \odot \mathbf{g} \tag{7.2}$$

$$\mathbf{h}_p = \mathbf{o} \odot \tanh(\mathbf{c}_p) \tag{7.3}$$

where $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias vector, σ is the sigmoid activation function, and $\odot$ is the element-wise product.

However, the Gumbel tree-LSTM differs from standard tree-LSTMs in that the selection of nodes to merge (i.e. the selection of left and right children) is done in an unsupervised manner at each timestep. At each timestep, each pair of adjacent nodes is considered for merging, and the hidden states $\widehat{\mathbf{h}}_i$ for each candidate parent representation are computed using equation 7.3. A composition query vector $\mathbf{q}$, which is simply a vector of trainable weights, is used to obtain a score v_i for each candidate representation as follows:

$$v_i = \frac{\exp\left(\mathbf{q} \cdot \widehat{\mathbf{h}}_i\right)}{\sum_j \exp\left(\mathbf{q} \cdot \widehat{\mathbf{h}}_j\right)} \tag{7.4}$$

Finally, the straight-through Gumbel softmax estimator (Jang et al., 2017) is used to sample a parent from the candidates $\widehat{\mathbf{h}}_i$ based on these scores v_i; this allows us to sample a discrete parent selection while still maintaining differentiability. The straight-through Gumbel softmax estimator is calculated by first sampling Gumbel noise g_i for each candidate:

$$g_i = -\log(-\log(u_i)) \tag{7.5}$$

where u_i is uniform over $(0,1)$. This noise is then used to calculate the Gumbel softmax over the scores v_i:

$$y_i = \frac{exp((log(v_i) + g_i)/\tau)}{\sum_j exp((log(v_j) + g_j)/\tau)} \tag{7.6}$$

where τ is a hyperparameter that models the temperature. In the forward pass, we take the argmax over y_i in order to make a discrete selection of a single parent state from the candidates $\widehat{\mathbf{h}}_i$, whereas in the backward pass, the continuous, differentiable Gumbel softmax from equation 7.6 is used.

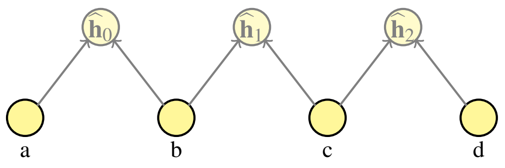
This process continues until there is only one remaining node that summarizes the entire sentence; we refer to this as the *root node*. The whole process is illustrated in Figure 7.1.

At inference time, straight-through Gumbel softmax is not used. Instead, we greedily select the highest-scoring candidate phrase.

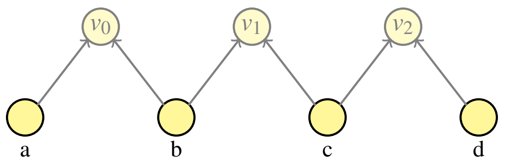
This Gumbel tree-LSTM encoder induces a binary hierarchy over the source sentence. For a sentence of length n, there are n word nodes and $n-1$ phrase nodes (including the root node). We initialize the decoder using the root node; attention to word and/or phrase nodes is described in section 7.2.4.

7.2.3 Top-Down Encoder Pass

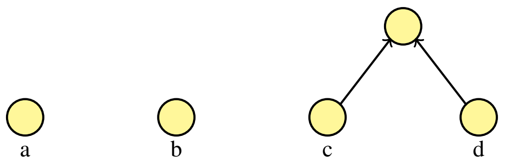
In the bottom-up tree-LSTM encoder described in the previous section, each node is able to incorporate local information from its respective children; however, no global information is used. Thus, we introduce a top-down pass, which allows the nodes to take global information about the entire tree into account. We refer to models containing this top-down pass as *top-down tree2seq* models. Adding a top-down pass has



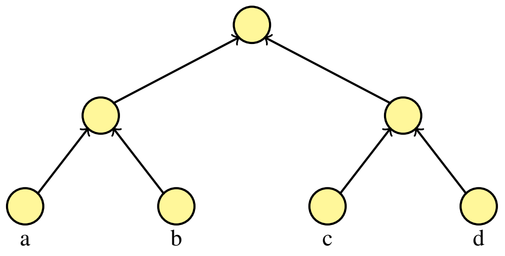
(a) Consider merging each pair of adjacent nodes. Use equations 7.1, 7.2, and 7.3 to calculate the hidden state $\widehat{\mathbf{h}}_i$ for each phrase candidate.



(b) Score each phrase candidate using equation 7.4.



(c) Select phrase node based on scores using straight-through Gumbel softmax estimator (by taking the argmax over equation 7.6).



(d) Continue the process until only a single root node remains.

Figure 7.1: Unsupervised induction of phrase nodes in the Gumbel tree-LSTM.

been shown to aid in tree-based NMT with supervised syntactic information (Chen et al., 2017; Yang et al., 2017); here, we add it to our unsupervised hierarchies.

Our top-down tree implementation is similar to the bidirectional tree-GRU introduced by Kokkinos and Potamianos (2017). The root node from the bottom-up pass already contains information about the entire tree; therefore, we define the top-down root node $\mathbf{h}^{\downarrow}_{root}$ as follows:

$$\mathbf{h}^{\downarrow}_{root} = \mathbf{h}^{\uparrow}_{root} \tag{7.7}$$

where $\mathbf{h}^{\uparrow}_{root}$ is the hidden state of the bottom-up root node calculated using the Gumbel tree-LSTM described in section 7.2.2.

For each remaining node, including word nodes, the top-down representation $\mathbf{h}^{\downarrow}_{i}$ is computed from its bottom-up hidden state representation $\mathbf{h}^{\uparrow}_{i}$ (calculated using the Gumbel tree-LSTM) and the top-down representation of its parent $\mathbf{h}^{\downarrow}_{p}$ (calculated during the previous top-down steps) using a GRU:

$$\begin{bmatrix} \mathbf{z}^{\downarrow}_{i} \\ \mathbf{r}^{\downarrow}_{i} \end{bmatrix} = \sigma\left(\mathbf{W}^{td}\mathbf{h}^{\uparrow}_{i} + \mathbf{U}^{td}\mathbf{h}^{\downarrow}_{p} + \mathbf{b}^{td}\right) \tag{7.8}$$

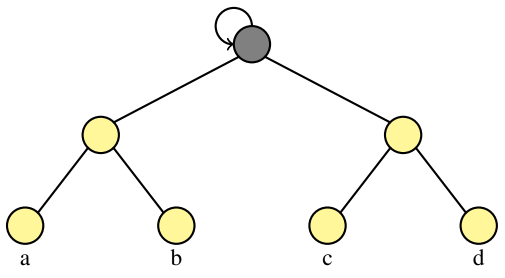
$$\widetilde{\mathbf{h}}^{\downarrow}_{i} = \tanh\left(\mathbf{W}^{td}_{h}\mathbf{h}^{\uparrow}_{i} + \mathbf{U}^{td}_{h}\left(\mathbf{r}^{\downarrow}_{i} \odot \mathbf{h}^{\downarrow}_{p}\right) + \mathbf{b}^{td}_{h}\right) \tag{7.9}$$

$$\mathbf{h}^{\downarrow}_{i} = \left(1 - \mathbf{z}^{\downarrow}_{i}\right)\mathbf{h}^{\downarrow}_{p} + \mathbf{z}^{\downarrow}_{i}\widetilde{\mathbf{h}}^{\downarrow}_{i} \tag{7.10}$$

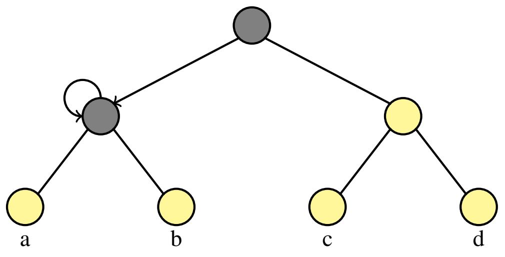
where $\mathbf{W}^{td}$, $\mathbf{U}^{td}$, $\mathbf{W}^{td}_{h}$, and $\mathbf{U}^{td}_{h}$ are weight matrices; $\mathbf{b}^{td}$ and $\mathbf{b}^{td}_{h}$ are bias vectors; and σ is the sigmoid activation function. We do not use different weights for the left and right children of a given parent. The entire top-down encoder pass is illustrated in Figure 7.2.

Each node needs a final representation to supply to the attention mechanism. Here, the top-down version of the node is used, because the top-down version captures both local and global information about the node.

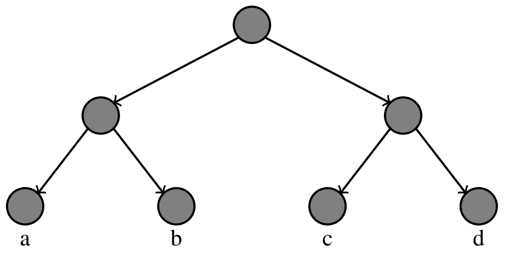
The decoder is initialized with the top-down representation of the root node. However, this is identical to the bottom-up representation of the root node, so no additional top-down information is used to initialize the decoder. Since the root node contains information about the entire sentence, this allows the decoder to be initialized with a summary of the source sentence, mirroring standard sequential NMT.



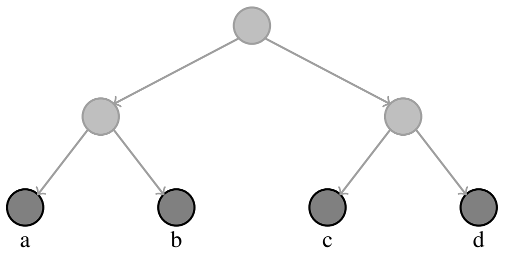
(a) Top-down root node is identical to its bottom-up version.



(b) Remaining nodes are calculated from their bottom-up version and their parents' top-down version.



(c) Process continues recursively until top-down word nodes are calculated.



(d) Once the top-down pass is completed, the phrase nodes are discarded and the decoder attends only to the word nodes.

Figure 7.2: Calculation of the top-down pass in the unsupervised hierarchical encoder.

Language Pair	Sentences
TL↔EN	50 962
TR↔EN	207 373
RO↔EN	608 320

Table 7.1: Amount of parallel training sentences for each language pair after preprocessing.

7.2.4 Attention to Words and Phrases

The bottom-up (section 7.2.2) and top-down (section 7.2.3) tree2seq models take different approaches to attention. The bottom-up model attends to the intermediate phrase nodes of the tree-LSTM, in addition to the word nodes output by the leaf LSTM. This follows what was done by Eriguchi et al. (2016). We use one attention mechanism for all nodes (word and phrase), making no distinction between different node types.

When the top-down pass (section 7.2.3) is added to the encoder, the final word nodes contain hierarchical information from the entire tree, as well as sequential information. Therefore, in the top-down tree2seq model, we attend to the top-down word nodes only, ignoring the phrase nodes. The idea behind this is that attention to the phrase nodes is unnecessary since the word nodes summarize the phrase-level information. Indeed, in preliminary experiments, attending to phrase nodes did not yield any improvements in translation performance.

7.3 Experimental Setup

7.3.1 Data

The models are evaluated on Tagalog (TL)↔English (EN), Turkish (TR)↔EN, and Romanian (RO)↔EN translation. These language pairs were selected because they range from very low-resource to medium-resource, so we can evaluate the proposed models at various settings. Table 7.1 displays the number of parallel training sentences for each language pair.

The TR↔EN and RO↔EN corpora are from the WMT16 shared task (Bojar et al., 2016). The models are validated on newsdev2016 and evaluated on newstest2016. The TL↔EN data is from IARPA MATERIAL Program language collection release IARPA_MATERIAL_BASE-1B-BUILD_v1.0. We do not use any monolingual data during training.

The data is tokenized and truecased with the Moses scripts (Koehn et al., 2007). We use byte pair encoding (Sennrich et al., 2016d) with 45k merge operations to split words into subwords. Notably, this means that the unsupervised tree encoder induces a binary parse tree over subwords (rather than at the word level).

7.3.2 Implementation and Training

All models are implemented in OpenNMT-py (Klein et al., 2017). They use word embedding size 500, hidden layer size 1000, batch size 64, two layers in the encoder and decoder, and dropout rate 0.3 (Gal and Ghahramani, 2016). We set maximum sentence length to 50 (150 for the source side of the parse2seq baseline). Models are trained using Adam (Kingma and Ba, 2015) with learning rate 0.001.

For tree-based models, we use a Gumbel temperature of 0.5, which performed best in preliminary experiments. The tree-LSTM component of the unsupervised tree2seq encoders has only a single layer.

We train until convergence on the validation set, and the model with the highest BLEU on the validation set is used to translate the test data. During inference, we set beam size to 12 and maximum length to 100.

7.3.3 Baselines

Seq2seq

We compare our models to an LSTM-based attentional NMT model; we refer to this model as *seq2seq*. Apart from the encoder, this baseline is identical to our proposed models. We train the seq2seq baseline on unparsed parallel data.

Parse2seq

For translations out of English, we also consider a baseline that uses syntactic supervision; we dub this model *parse2seq*. This parse2seq baseline is identical to the parse2seq model described in section 5.4.3. We only apply this baseline to translations out of English because we only have a constituency parser for English (Stanford CoreNLP; Manning et al., 2014).

BLEU	**TL→EN**	**TR→EN**	**RO→EN**
seq2seq	17.9	11.1	**29.3**
bottom-up tree2seq	**26.1**	12.8	28.6
top-down tree2seq	25.3	**13.2**	28.6

Table 7.2: BLEU for the baseline and the unsupervised tree2seq systems on *→EN translation.

BLEU	**EN→TL**	**EN→TR**	**EN→RO**
seq2seq	15.9	8.5	27.3
parse2seq	17.1	9.0	**28.4**
bottom-up tree2seq	**23.1**	9.7	27.3
top-down tree2seq	22.5	**9.8**	27.0

Table 7.3: BLEU for the baselines and the unsupervised tree2seq systems on EN→* translation.

7.4 Results

7.4.1 Main Experiments

Tables 7.2 and 7.3 display translation performance as estimated by BLEU scores for our unsupervised tree2seq models translating into and out of English, respectively. For the lower-resource language pairs, TL↔EN and TR↔EN, both tree2seq models consistently improve over the seq2seq and parse2seq baselines. However, for the medium-resource language pair (RO↔EN), the unsupervised tree models do not improve over seq2seq, whereas the parse2seq baseline does. These results indicate that inducing hierarchies on the source side is very helpful in low-resource scenarios, but the utility of this method decreases as more data becomes available.

In Table 7.4, we display the number of parameters for each system on translation from EN (translation into EN is similar). Both proposed tree2seq systems have significantly more parameters than the baseline systems, which may be an explanation for the improvements on TL↔EN and TR↔EN translation. Additionally, the top-down tree2seq model has roughly six million more parameters than the bottom-up version; however, despite having more parameters, the top-down model does not consistently outperform the bottom-up model.

Although the tree2seq models improve over the baselines for low- and very low-

Parameters	**EN→TL**	**EN→TR**	**EN→RO**
seq2seq	94M	93M	95M
parse2seq	94M	93M	95M
bottom-up tree2seq	102M	101M	103M
top-down tree2seq	108M	107M	109M

Table 7.4: Number of parameters for each model trained on EN→* translation. Number of parameters for *→EN is similar.

resource translation, there are some trade-offs in terms of training time. Compared to the seq2seq baseline, the bottom-up and top-down tree2seq models take roughly twice as long to train, as does the parse2seq baseline. Additionally, inference using the parse2seq and tree2seq models takes slightly longer than for the seq2seq model.

7.4.2 Analysis

Unsupervised Parses

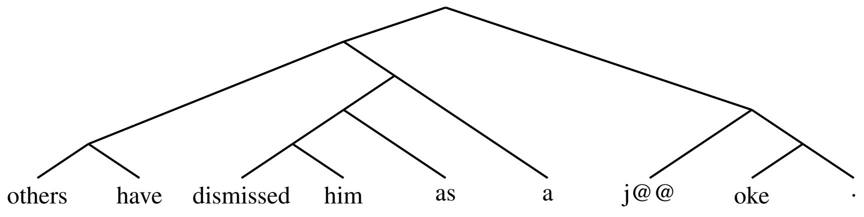
Williams et al. (2018) observed that the parses resulting from Gumbel tree-LSTMs for sentence classification did not seem to fit a known formalism. An examination of the parses induced by our NMT models suggests this as well. Furthermore, the different architectures (bottom-up and top-down tree2seq) do not seem to learn the same parses for the same language pair, nor does the same architecture learn the same parses for different language pairs. Figure 7.3 displays examples of parses induced by the trained systems on a sentence from the test data.

Subword Recombination

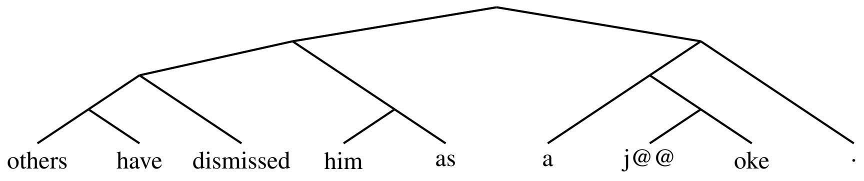
The unsupervised parses are trained over subwords; if the induced hierarchies have a linguistic basis, it would be reasonable to expect the model to combine subwords into words as a first step. Figure 7.4 illustrates the expected and unexpected subword combinations from a linguistic point of view.

In order to evaluate whether the models have a notion of words, we calculate the percentage of subwords that are recombined correctly for each model; these results are in Table 7.5. We also include results for right-branching and randomly combined trees for contextualization.[1]

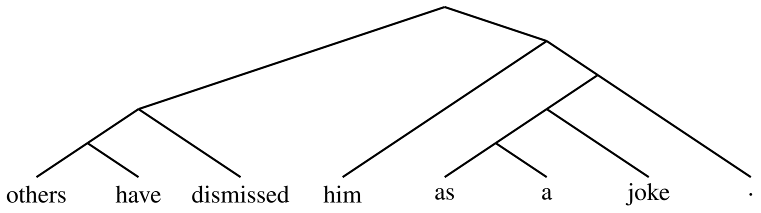
[1]The right-branching subword combination accuracy is very low because only the last two units in a



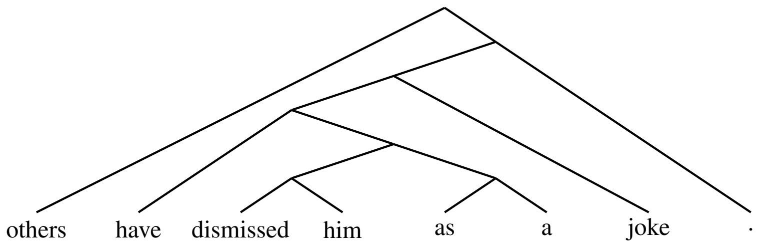
(a) EN→TR bottom-up.



(b) EN→TR top-down.

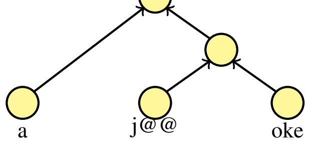


(c) EN→RO bottom-up.

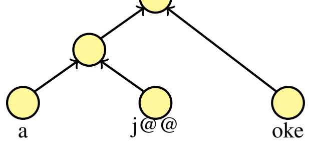


(d) EN→RO top-down.

Figure 7.3: Induced parses on an example sentence from the test data.



(a) Expected subword combination.



(b) Unexpected subword combination.

Figure 7.4: Expected and unexpected subword combination for the phrase *a j@@ oke*.

Language Pair	right branch	random	bottom-up	top-down
EN→TL	0.0%	39.8%	22.1%	16.4%
EN→TR	0.8%	34.7%	29.3%	21.3%
EN→RO	0.9%	34.3%	27.2%	27.2%
TL→EN	0.2%	32.7%	12.7%	22.7%
TR→EN	0.4%	34.2%	27.7%	22.9%
RO→EN	0.7%	34.0%	30.8%	11.4%

Table 7.5: Recombined subwords in the test data. We include results for our proposed bottom-up and top-down tree2seq models, as well as for right-branching and randomly combined subwords.

Corroborating the observations in the previous section, only a very low percentage of subwords is correctly recombined for each tree2seq model. For all language pairs, the percent of correctly recombined subwords for both proposed models is much lower than for the random parse trees. This gives further indication that the parses the model learns do not seem to be based on linguistics. In addition, subword recombination accuracy does not seem to correlate with translation performance (as shown in Tables 7.2 and 7.3).

7.5 Discussion

Our unsupervised tree2seq models leverage a hierarchical encoder to add structural information to neural machine translation. For the low-resource language pairs EN↔TR and EN↔TL, the proposed models achieve strong improvements in BLEU over the syntactic and non-syntactic baselines. However, for medium-resource EN↔RO, standard seq2seq does better than our unsupervised tree2seq. Our two proposed models, bottom-up and top-down tree2seq, did not differ much in translation performance, so it seems that adding the top-down pass to incorporate global sentence information is not particularly helpful.

Our unsupervised tree2seq models have several advantages in addition to the improvements shown in low-resource NMT. Since they do not rely on a syntactic parser, they can be applied to any source language, including a very low-resource language without outside resources. These models are also very flexible and could easily be

sentence can be directly combined in a right-branching tree.

augmented with monolingual target data or some syntactic supervision, since the encoder is able to induce a hierarchy over any sequence of tokens. These benefits do not come at a large training cost, since the use of the straight-through Gumbel softmax estimator (Jang et al., 2017) allows our model to induce discrete parses while still being fully differentiable.

The proposed models have the disadvantage that they do not work well for translating medium-resource language pairs; when more parallel data is available, it is better to use a standard seq2seq or syntactic NMT model (such as those proposed in chapters 5 and 6). They are also not as easily transferable to new architectures as our models in chapters 5 and 6 that used linearized parses. Additionally, as discussed in section 7.4.2, they do not seem to learn any consistent linguistic information. Finally, although we show improvements over our baselines for low- and very low-resource language pairs, these models do not improve in BLEU over the state of the art, since we did not use back-translation (Sennrich et al., 2016c), a copied corpus (chapter 3), or ensembles of multiple models. However, we expect that the improvements from using our unsupervised tree2seq model might be orthogonal to the gains that can be made through monolingual data and multiple models.

7.6 Subsequent Work

Concurrent or subsequent papers to the work in this chapter have further studied the Gumbel tree-LSTM and the use of hierarchical structure in neural machine translation. Here, we give a brief overview of these papers.

Williams et al. (2018) examined the latent tree structures learned by Gumbel tree-LSTMs (and other models) trained on a textual entailment task. Their results mirrored the results in this chapter. First, they found that the Gumbel tree-LSTM outperformed other models (including syntactic and LSTM baselines) on the main textual entailment task. They also found that the Gumbel tree-LSTM did not produce consistent trees across different training runs, and that the grammar learned by the Gumbel tree-LSTM did not resemble Penn Treebank grammar.

Shi et al. (2018) studied various tree-based encoders for a variety of tasks, including moderate-resource neural machine translation. On NMT, they found that right-branching trees resulted in the highest BLEU scores. The vanilla Gumbel tree-LSTM encoder performed similarly to parse trees and to balanced binary trees, while the left-branching trees and the standard LSTM performed worst. They argued that right-branching trees' strong performance is due to the fact that the leftmost words are closest to the final sentence representation (this is similar to the reversal of the source sentence in NMT done by Sutskever et al., 2014).

7.7 Conclusions

In this chapter, we proposed a method for incorporating unsupervised structure into the source side of neural machine translation. For low-resource language pairs, this method yielded strong improvements over both sequential and parsed baselines. This technique is useful for adding hierarchical structure into low-resource NMT when a source-language parser is not available. Further analysis indicated that the structures induced by the proposed encoder are not similar to known linguistic structures, corroborating other work that has analyzed these models. Overall, we found that unsupervised source structure was beneficial for low-resource cases, while supervised source structure (i.e. parses of the source sentence) was beneficial for medium-resource neural machine translation.

In this thesis, we have proposed ways of incorporating both monolingual and syntactic data into neural machine translation training. This chapter builds on the syntactic aspect by considering the case of low-resource NMT. Low-resource neural machine translation is a task where we expect non-parallel data to be particularly helpful; however, for low-resource source languages, we may not have a parser or even a treebank, so it can be hard to add syntactic information into NMT in these cases. This chapter addresses this issue by inducing an unsupervised hierarchical structure over source sentences; in very low-resource cases, the induced structure actually outperforms supervised syntactic parses. Thus, this chapter can be considered a complement to the previous two syntactic NMT chapters.

There are several potential extensions of this work that we hope to address in the future. First, we would like to explore ways of inducing unsupervised hierarchies in the decoder; such work would build on Wang et al. (2018). To our knowledge, using hierarchies on both the encoder and the decoder has not yet been studied, so this could be an interesting area for future work. Additionally, we would like to try adding some syntactic supervision to the source trees, for example in the form of pre-training on parses, in order to see whether actual syntactic parses can improve our models.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

The goal of this thesis has been to explore ways of using non-traditional data to improve neural machine translation, where we defined non-traditional data as anything besides sentence-aligned parallel data in the source and target languages. This is an important task because while neural machine translation performs well when large amounts of parallel data are available (Hassan et al., 2018; Wu et al., 2016), large amounts of parallel data are unavailable for all but the highest-resource language pairs. Thus, it is necessary to find ways to reduce NMT's reliance on parallel data by adding in other types of data. This thesis has been a step in that direction.

We have taken two main approaches to adding non-traditional data to neural machine translation. The first is using monolingual data in addition to the parallel training data. Unlike parallel data, monolingual data is relatively abundant for many languages, so it is an important source of outside training data for NMT. We have concentrated on using monolingual target-language data and monolingual data in a pivot language (i.e. neither the source nor the target language), and briefly looked at using source monolingual data as well. Our second source of non-traditional data has been syntactic annotations. We have mainly worked on translating from English using additional information about English syntax, which has proven to be helpful particularly in low-resource scenarios. Although we focused on translating from a high-resource language (English), we also presented a method for incorporating source-side hierarchies without requiring syntactic annotations; this method can be applied to any source language and was beneficial in very low-resource scenarios.

Chapters 3 and 4 addressed injecting monolingual data into neural machine translation. In chapter 3, we introduced a straightforward method for adding target-language monolingual data into NMT. This method consisted of converting a monolingual corpus into a bitext by copying it, so that the source and target sides of the text were identical. This pseudo-parallel corpus was then mixed with the parallel training data in order to train the NMT system. This method had the advantage that it did not require a potentially time-consuming pre-training step, unlike the popular back-translation method of Sennrich et al. (2016c). In addition, for low-resource language pairs, the copied monolingual corpus could also be combined with a back-translated corpus in order to achieve further improvements in BLEU. Further analysis indicated that this copied monolingual corpus helped the model learn to copy words that were identical in the source and target text. However, this method had some disadvantages, as well. First, it did not improve translation performance for a high-resource language pair (English↔German), although it did not damage the translation performance either. Second, the copied corpus alone did not outperform back-translation alone, although it did outperform a baseline without monolingual data. Finally, we tried to use the copied corpus system in order to incorporate source-side monolingual data in addition to the target data, but we did not see any improvements for this setup, even for a low-resource language pair. To summarize, the copied corpus method was very effective for adding target monolingual data into low- and medium-resource neural machine translation, with or without back-translation, but it was less effective for source monolingual data or high-resource NMT.

Our second proposal for adding monolingual data into NMT was presented in chapter 4. In that chapter, we considered zero-shot and zero-resource NMT, where no direct source $\leftrightarrow$ target parallel data is available but parallel data with a pivot language is used. We introduced a novel type of data for this task: monolingual data in the pivot language. In many cases, the pivot language is the highest-resource language of the three (e.g. it is often English), so we can expect it to have the most high-quality monolingual data; thus, it is logical to choose the pivot language as the language of the monolingual corpus. However, while monolingual source and target data could easily be added through an extension of bidirectional NMT with back-translation (Niu et al., 2018), it is not immediately clear how to apply monolingual pivot-language data to this task. Our proposal was to use the initial zero-shot system to translate the pivot monolingual data into both the source and the target language, and then to use the resulting source' $\rightarrow$ target' and target' $\rightarrow$ source' parallel corpora to fine-tune or re-train the model. We found that doing so improved over zero-shot and zero-resource baselines, as well as over back-translation and the copied corpus method. Our methods for using pivot monolingual data were also successful at direct translation, so inference was not costlier than for standard NMT. The main disadvantage of this method was that it required a source $\leftrightarrow$ target NMT system be trained specifically for this task, so it would not be usable in a single massively multilingual system that translates between several languages (although such a multilingual system could be used as a starting point for our method).

In chapters 5, 6, and 7, we turned our attention from monolingual corpora to syntactic annotations as our type of non-parallel data. Chapter 5 proposed a multi-source method for adding source syntax into a high-resource NMT system that took both parsed and unparsed source sentences as simultaneous input. This model had several advantages over previous syntactic NMT work: it would be easily applicable to any architecture (LSTMs, transformers, CNNs, etc.), it was able to translate reasonably well from unparsed sentences, and it could be trained with a mix of unparsed and parsed data. Additionally, the proposed model improved in BLEU over both syntactic and non-syntactic baselines. We further found that the multi-source model was better at translating long sentences than the baselines, but it was a bit slower in both training and inference when parsed sentences were used (due to the time it took to parse the sentences). Finally, we introduced a shared encoder method that trained a single NMT model to translate from a source sentence in either its parsed or unparsed versions; this model also strongly outperformed the baselines. However, we showed in chapter 6 that these improvements were not as large when the shared encoder model was used in a stronger transformer system.

Chapter 6 built on the work in chapter 5 by adding syntax into the transformer model. The multi-task model proposed in this chapter, like the multi-source and shared encoder models, used linearized parses to inject source syntax into NMT. We used a multi-task parsing and translation framework to leverage these linearized parses; this model had the advantage that it could be used with different architectures and did not require parsed data at inference time. We evaluated our multi-task model on several diverse target languages for low-, medium-, and high-resource data scenarios; this was to our knowledge the first work to perform a comprehensive cross-linguistic evaluation of syntactic NMT. We found that the multi-task model aided translation for low-resource scenarios whereas it was not helpful for high-resource cases. Further analysis showed that language family seemed to have an effect on parsing performance, but not on translation performance. The main limitation of this model was its inability to aid high-resource NMT; it may be that a more elaborate scheduling regime (Kiperwasser and Ballesteros, 2018) would have improved performance in these cases.

We showed in chapters 5 and 6 that source-side syntax could improve neural machine translation performance in low-resource scenarios. However, when translating out of a low-resource language, syntactic annotation might not always be available. Thus, in chapter 7, we introduced an unsupervised tree2seq model that used a Gumbel tree-LSTM (Choi et al., 2018) encoder to induce unsupervised hierarchies on the source sentences. This model was very successful in improving low-resource NMT performance, without requiring significantly more training or inference time. However, for medium-resource cases, it did not do any better than a vanilla seq2seq model. In addition, the hierarchies that were induced over the source sentences did not seem to have a linguistic basis, and the models did not learn to combine subwords into words consistently.

8.2 Future Work

In this thesis, we have explored leveraging monolingual and syntactic information to augment the parallel corpora typically used in neural machine translation. There are several other options for adding non-parallel data to NMT, as well. Here, we give some ideas for future work on the topic of using non-parallel data in neural machine translation.

- **Adding monolingual data to neural machine translation** is an important area for future work because of the ubiquity of monolingual data (compared to parallel data). In this thesis, we have looked at adding monolingual target-language (chapter 3) and pivot-language (chapter 4) corpora to NMT training. Other options to explore in the future include adding source-language data, data in multiple pivot languages, or data in a language related to the source or target language; various types of monolingual data could also be added simultaneously. Additionally, unsupervised machine translation (Artetxe et al., 2018; Lample et al., 2018) has recently been shown to be viable; we would like to see whether the methods proposed in this thesis could further improve the unsupervised machine translation models.

- **Improving syntax-based neural machine translation** is an interesting future direction. First, we would like to extend the multi-source and multi-task models proposed in chapters 5 and 6 to the target side, in order to add target syntax into NMT as well. This would be a relatively straightforward extension of these models; we could even add source and target syntax simultaneously for high-resource language pairs. It would also be interesting to add an unsupervised hierarchical decoder to match the unsupervised hierarchical encoder proposed in chapter 7; however, there is some indication in prior literature that this may not be successful (Wang et al., 2018). On the topic of the unsupervised hierarchies, we would also like to see whether it is possible to add semi-supervised syntax, for example by pre-training the hierarchies on gold syntactic data. Finally, current research does not have a thorough understanding of when and why syntactic annotations can help NMT. More analysis is needed to better understand these models in order to make them useful to apply in practice.

- **Combining monolingual corpora and syntactic annotations** will be important in order to build state-of-the-art systems. Throughout this thesis, we have evaluated each of our proposed improvements separately. However, for our syntactic NMT models in particular, we have taken care to ensure that it would be straightforward to add pseudo-parallel corpora created using monolingual data into the systems. In the future, we would like to combine the methods proposed here and see whether they successfully stack together. Additionally, to our knowledge, there has been little research on combining syntactic NMT with other techniques, such as zero-shot NMT or unsupervised machine translation.

- **Other types of non-parallel data** are also available; in the future, a natural extension to this work would be to study adding such data to further improve neural machine translation. One of the most ubiquitous types of data is out-of-domain (parallel or monolingual) corpora. Neural machine translation does well when trained on in-domain data, but it can do much worse than statistical machine translation when tested on a domain different from the one it was trained on (Koehn and Knowles, 2017). Thus, domain adaptation is an important problem to solve in order to be able to incorporate all available data into NMT training. Another source of non-traditional training data for neural machine translation is data from related languages. This can be helpful in cases where a low-resource language is closely related to a high-resource language. Currently, most research done on this topic uses multilingual NMT models directly, but there may be better ways to leverage the data when some languages are very closely related. Finally, multi-modal information, such as images and speech data, is becoming popular in neural machine translation research; in the future, we would like to extend our work to multi-modal neural machine translation as well.

Bibliography

Aharoni, R. and Goldberg, Y. (2017). Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 132–140. Association for Computational Linguistics.

Aharoni, R., Johnson, M., and Firat, O. (2019). Massively multilingual neural machine translation. *arXiv preprint arXiv:1903.00089*.

Arivazhagan, N., Bapna, A., Firat, O., Aharoni, R., Johnson, M., and Macherey, W. (2019). The missing ingredient in zero-shot neural machine translation. *arXiv preprint arXiv:1903.07091*.

Artetxe, M., Labaka, G., Agirre, E., and Cho, K. (2018). Unsupervised neural machine translation. In *6th International Conference on Learning Representations*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*.

Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., and Sima'an, K. (2017). Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967. Association for Computational Linguistics.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2016). Neural versus phrase-based machine translation quality: A case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267. Association for Computational Linguistics.

Bertoldi, N. and Federico, M. (2009). Domain adaptation for statistical machine translation with monolingual resources. In *Proceedings of the Fourth Workshop on Sta-*

tistical Machine Translation, pages 182–189. Association for Computational Linguistics.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C., Negri, M., Post, M., Rubino, R., Specia, L., and Turchi, M. (2017). Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214. Association for Computational Linguistics.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., Logacheva, V., Monz, C., Negri, M., Névéol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., and Zampieri, M. (2016). Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198. Association for Computational Linguistics.

Bojar, O., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Koehn, P., and Monz, C. (2018). Findings of the 2018 Conference on Machine Translation (WMT18). In *Proceedings of the Third Conference on Machine Translation*, pages 272–303. Association for Computational Linguistics.

Bojar, O. and Tamchyna, A. (2011). Improving translation model by monolingual data. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 330–336. Association for Computational Linguistics.

Castilho, S., Moorkens, J., Gaspari, F., Sennrich, R., Sosoni, V., Georgakopoulou, Y., Lohar, P., Way, A., Miceli Barone, A. V., and Gialama, M. (2017). A comparative quality evaluation of PBSMT and NMT using professional translators. In *Proceedings of Machine Translation Summit XVI*, pages 116–131.

Chen, H., Huang, S., Chiang, D., and Chen, J. (2017). Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 1936–1945. Association for Computational Linguistics.

Cheng, Y., Xu, W., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016). Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the ACL*, pages 1965–1974. Association for Computational Linguistics.

Cheng, Y., Yang, Q., Liu, Y., Sun, M., and Xu, W. (2017). Joint training for pivot-based neural machine translation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3974–3980.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734. Association for Computational Linguistics.

Choe, D. K. and Charniak, E. (2016). Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336. Association for Computational Linguistics.

Choi, J., Yoo, K. M., and Lee, S.-g. (2018). Learning to compose task-specific tree structures. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2018). Word translation without parallel data. In *6th International Conference on Learning Representations*.

Cotterell, R., Mielke, S. J., Eisner, J., and Roark, B. (2018). Are all languages equally hard to language-model? In *Proceedings of NAACL-HLT*, pages 536–541. Association for Computational Linguistics.

Currey, A. and Heafield, K. (2018a). Multi-source syntactic neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2961–2966. Association for Computational Linguistics.

Currey, A. and Heafield, K. (2018b). Unsupervised source hierarchies for low-resource neural machine translation. In *Proceedings of the Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP*, pages 6–12. Association for Computational Linguistics.

Currey, A. and Heafield, K. (2019). Incorporating source syntax into transformer-based neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation*. Association for Computational Linguistics.

Currey, A., Miceli Barone, A. V., and Heafield, K. (2017). Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156. Association for Computational Linguistics.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing*, pages 1723–1732. Association for Computational Linguistics.

Edunov, S., Ott, M., Auli, M., and Grangier, D. (2018). Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500. Association for Computational Linguistics.

Eriguchi, A., Hashimoto, K., and Tsuruoka, Y. (2016). Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the ACL*, pages 823–833. Association for Computational Linguistics.

Firat, O., Cho, K., and Bengio, Y. (2016a). Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of NAACL-HLT*, pages 866–875. Association for Computational Linguistics.

Firat, O., Sankaran, B., Al-Onaizan, Y., Yarman Vural, F. T., and Cho, K. (2016b). Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277. Association for Computational Linguistics.

Gal, Y. and Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29*, pages 1019–1027.

Gehring, J., Auli, M., Grangier, D., and Dauphin, Y. N. (2017a). A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 123–135. Association for Computational Linguistics.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017b). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1243–1252.

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

Guzmán, F., Chen, P.-J., Ott, M., Pino, J., Lample, G., Koehn, P., Chaudhary, V., and Ranzato, M. (2019). Two new evaluation datasets for low-resource machine translation: Nepali-English and Sinhala-English. *arXiv preprint arXiv:1902.01382*.

Ha, T.-L., Niehues, J., and Waibel, A. (2016). Toward multilingual neural machine translation with universal encoder and decoder. In *Proceedings of the 13th International Workshop on Spoken Language Translation*.

Hashimoto, K. and Tsuruoka, Y. (2017). Neural machine translation with source-side latent graph parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 125–135. Association for Computational Linguistics.

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J., Federmann, C., Huang, X., Junczys-Dowmunt, M., Lewis, W., Li, M., Liu, S., Liu, T.-Y., Luo, R., Menezes, A., Qin, T., Seide, F., Tan, X., Tian, F., Wu, L., Wu, S., Xia, Y., Zhang, D., Zhang, Z., and Zhou, M. (2018). Achieving human parity on automatic Chinese to English news translation. *arXiv preprint arXiv:1803.05567*.

He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T., and Ma, W.-Y. (2016). Dual learning for machine translation. In *Advances in Neural Information Processing Systems 29*, pages 820–828.

Heafield, K. (2011). KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics.

Helcl, J. and Libovickỳ, J. (2017). Neural Monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics*, 107(1):5–17.

Hieber, F., Domhan, T., Denkowski, M., Vilar, D., Sokolov, A., Clifton, A., and Post, M. (2017). Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*.

Hoang, V. C. D., Koehn, P., Haffari, G., and Cohn, T. (2018). Iterative back-translation for neural machine translation. In *Proceedings of the Second Workshop on Neural Machine Translation and Generation*, pages 18–24. Association for Computational Linguistics.

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

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with Gumbel-softmax. In *5th International Conference on Learning Representations*.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. Association for Computational Linguistics.

Khayrallah, H. and Koehn, P. (2018). On the impact of various types of noise on neural machine translation. In *Proceedings of the Second Workshop on Neural Machine Translation and Generation*, pages 74–83. Association for Computational Linguistics.

Kim, Y., Denton, C., Hoang, L., and Rush, A. M. (2017). Structured attention networks. In *5th International Conference on Learning Representations*.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Kiperwasser, E. and Ballesteros, M. (2018). Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the ACL*. Association for Computational Linguistics.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X*, volume 5, pages 79–86.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL*, pages 177–180. Association for Computational Linguistics.

Koehn, P. and Knowles, R. (2017). Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39. Association for Computational Linguistics.

Kokkinos, F. and Potamianos, A. (2017). Structural attention neural networks for improved sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 586–591. Association for Computational Linguistics.

Lakew, S. M., Lotito, Q. F., Negri, M., Turchi, M., and Federico, M. (2017). Improving zero-shot translation of low-resource languages. In *Proceedings of the 14th International Workshop on Spoken Language Translation*.

Lample, G., Denoyer, L., and Ranzato, M. (2018). Unsupervised machine translation using monolingual corpora only. In *6th International Conference on Learning Representations*.

Li, J., Xiong, D., Tu, Z., Zhu, M., Zhang, M., and Zhou, G. (2017). Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 688–697. Association for Computational Linguistics.

Libovickỳ, J. and Helcl, J. (2017). Attention strategies for multi-source sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 196–202. Association for Computational Linguistics.

Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Lu, Y., Keung, P., Ladhak, F., Bhardwaj, V., Zhang, S., and Sun, J. (2018). A neural interlingua for multilingual machine translation. In *Proceedings of the Third Conference on Machine Translation*, pages 84–92. Association for Computational Linguistics.

Luong, M.-T., Le, Q. V., Sutskever, I., Vinyals, O., and Kaiser, L. (2016). Multitask sequence to sequence learning. In *4th International Conference on Learning Representations*.

Luong, M.-T., Pham, H., and Manning, C. D. (2015a). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2015b). Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing*, pages 11–19. Association for Computational Linguistics.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the ACL*, pages 55–60. Association for Computational Linguistics.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Nadejde, M., Reddy, S., Sennrich, R., Dwojak, T., Junczys-Dowmunt, M., Koehn, P., and Birch, A. (2017). Predicting target language CCG supertags improves neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 68–79. Association for Computational Linguistics.

Niu, X., Denkowski, M., and Carpuat, M. (2018). Bi-directional neural machine translation with synthetic parallel data. In *Proceedings of the Second Workshop on Neural*

Machine Translation and Generation, pages 84–91. Association for Computational Linguistics.

Niu, X., Xu, W., and Carpuat, M. (2019). Bi-directional differentiable input reconstruction for low-resource neural machine translation. In *Proceedings of NAACL*, pages 442–448. Association for Computational Linguistics.

Ott, M., Auli, M., Grangier, D., et al. (2018). Analyzing uncertainty in neural machine translation. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3953–3962.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 311–318. Association for Computational Linguistics.

Park, J., Song, J., and Yoon, S. (2017). Building a neural machine translation system using only synthetic parallel data. *arXiv preprint arXiv:1704.00253*.

Platanios, E. A., Sachan, M., Neubig, G., and Mitchell, T. (2018). Contextual parameter generation for universal neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 425–435. Association for Computational Linguistics.

Raganato, A. and Tiedemann, J. (2018). An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297. Association for Computational Linguistics.

Ramachandran, P., Liu, P., and Le, Q. (2017). Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 383–391. Association for Computational Linguistics.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 1073–1083. Association for Computational Linguistics.

Sennrich, R. (2017). How grammatical is character-level neural machine translation? Assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th*

Conference of the European Chapter of the Association for Computational Linguistics, pages 376–382. Association for Computational Linguistics.

Sennrich, R., Birch, A., Currey, A., Germann, U., Haddow, B., Heafield, K., Miceli Barone, A. V., and Williams, P. (2017a). The University of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399. Association for Computational Linguistics.

Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., Läubli, S., Miceli Barone, A. V., Mokry, J., and Nadejde, M. (2017b). Nematus: A toolkit for neural machine translation. In *Proceedings of the EACL 2017 Software Demonstrations*, pages 65–68. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016a). Controlling politeness in neural machine translation via side constraints. In *Proceedings of NAACL-HLT*, pages 35–40.

Sennrich, R., Haddow, B., and Birch, A. (2016b). Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016c). Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the ACL*, pages 86–96. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016d). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the ACL*, pages 1715–1725. Association for Computational Linguistics.

Shi, H., Zhou, H., Chen, J., and Li, L. (2018). On tree-based neural sentence modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4631–4641. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Tai, K. S., Socher, R., and Manning, C. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd*

Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing, pages 1556–1566. Association for Computational Linguistics.

Tang, G., Müller, M., Rios, A., and Sennrich, R. (2018). Why self-attention? A targeted evaluation of neural machine translation architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4263–4272. Association for Computational Linguistics.

Tran, K., Bisazza, A., and Monz, C. (2018). The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4731–4736. Association for Computational Linguistics.

Tyers, F. M. and Alperen, M. S. (2010). South-East European Times: A parallel corpus of Balkan languages. In *Proceedings of the LREC Workshop on Exploitation of Multilingual Resources and Tools for Central and (South-) Eastern European Languages*, pages 49–53. European Language Resources Association.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Vinyals, O., Kaiser, Ł., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. (2015). Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28*, pages 2773–2781.

Wang, X., Pham, H., Yin, P., and Neubig, G. (2018). A tree-based decoder for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4772–4777. Association for Computational Linguistics.

Williams, A., Drozdov, A., and Bowman, S. R. (2018). Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., ukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yang, B., Wong, D. F., Xiao, T., Chao, L. S., and Zhu, J. (2017). Towards bidirectional hierarchical representations for attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1432–1441. Association for Computational Linguistics.

Yogatama, D., Blunsom, P., Dyer, C., Grefenstette, E., and Ling, W. (2017). Learning to compose words into sentences with reinforcement learning. In *5th International Conference on Learning Representations*.

Zaremoodi, P. and Haffari, G. (2018). Incorporating syntactic uncertainty in neural machine translation with a forest-to-sequence model. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1421–1429. Association for Computational Linguistics.

Zhang, J. and Zong, C. (2016). Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Zoph, B. and Knight, K. (2016). Multi-source neural translation. In *Proceedings of NAACL-HLT*, pages 30–34. Association for Computational Linguistics.