FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

ABSTRACT OF DOCTORAL THESIS

Jindřich Helcl

Non-Autoregressive Neural Machine Translation

Institute of Formal and Applied Linguistics

Supervisor:	prof. RNDr. Jan Hajič, Dr.
Study programme:	Computer Science
Study branch:	Mathematical Linguistics

Prague 2021

The results of this thesis were achieved in the period of a doctoral study at the Faculty of Mathematics and Physics, Charles University in years 2014–2021.

Student: Mgr. Jindřich Helcl

Supervisor: prof. RNDr. Jan Hajič, Dr.
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University
Malostranské nám. 25, 118 00 Prague 1

Department: Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University
Malostranské nám. 25, 118 00 Prague 1

Opponents: Kevin Duh, PhD.
Department of Computer Science
Center for Language and Speech Processing
Johns Hopkins University
HLTCOE Stieff Building / 810 Wyman Park Drive
Baltimore, MD 21211-2840, USA

Mgr. Martin Popel, Ph.D.
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University
Malostranské naměstí 25, 118 00 Prague 1, Czech Republic

The thesis defence will take place on February 9, 2022 at 2:00 p.m. in front of a committee for thesis defences in the branch Mathematical Linguistics at the Faculty of Mathematics and Physics, Charles University, Malostranské nám. 25, Prague 1, room S1.

Chairman of Academic Council: doc. Ing. Zdeněk Žabokrtský, Ph.D.
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University
Malostranské nám. 25, 118 00 Prague 1

The thesis can be viewed at the Study Department of Doctoral Studies of the Faculty of Mathematics and Physics, Charles University, Ke Karlovu 3, Prague 2.

This abstract was distributed on January 18, 2022

MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

AUTOREFERÁT DISERTAČNÍ PRÁCE

Jindřich Helcl

Neautoregresivní neuronový strojový překlad

Ústav formální a aplikované lingvistiky

Školitel:	prof. RNDr. Jan Hajič, Dr.
Studijní program:	Informatika
Studijní obor:	Matematická lingvistika

Praha 2021

Disertační práce byla vypracována na základě výsledků získaných během doktorského studia na Matematicko-fyzikální fakultě Univerzity Karlovy v letech 2014–2021.

Doktorand: Mgr. Jindřich Helcl

Školitel: prof. RNDr. Jan Hajič, Dr.
Ústav formální a aplikované lingvistiky
Matematicko-fyzikální fakulta
Univerzita Karlova
Malostranské nám. 25, 118 00 Praha 1

Školicí pracoviště: Ústav formální a aplikované lingvistiky
Matematicko-fyzikální fakulta
Univerzita Karlova
Malostranské nám. 25, 118 00 Praha 1

Oponenti: Kevin Duh, PhD.
Department of Computer Science
Center for Language and Speech Processing
Johns Hopkins University
HLTCOE Stieff Building / 810 Wyman Park Drive
Baltimore, MD 21211-2840, USA

Mgr. Martin Popel, Ph.D.
Ústav formální a aplikované lingvistiky
Matematicko-fyzikální fakulta
Univerzita Karlova
Malostranské naměstí 25, 118 00 Praha 1, Czech Republic

Obhajoba disertační práce se koná dne 9. února 2019 ve 14:00 před komisí pro obhajoby disertačních prací v oboru Matematická lingvistika na Matematicko-fyzikální fakultě UK, Malostranské nám. 25, Praha 1, v místnosti S1.

Předseda RDSO: doc. Ing. Zdeněk Žabokrtský, Ph.D.
Ústav formální a aplikované lingvistiky
Matematicko-fyzikální fakulta
Univerzita Karlova
Malostranské nám. 25, 118 00 Praha 1

S disertační prací je možno se seznámit na studijním oddělení Matematicko-fyzikální fakulty UK, Ke Karlovu 3, Praha 2.

Autoreferát byl rozeslán dne 18. ledna 2022.

Contents

1. Introduction

In real-world applications of machine translation (MT), efficiency is often crucial. Most commercial neural machine translation (NMT) models are available through a cloud-based service, such as Microsoft Translator[1] or Google Translate.[2] Scaling cloud-based solutions for large user bases is simple but costly. Even with a large pool of computational resources, it is worthwhile to implement optimizations that decrease model latency and improve user experience.

Locally deployed NMT models provide a number of advantages over cloud-based solutions. First, the service does not rely on the internet connection. Second, the data is not being sent to a third party server and, therefore, it is suitable for translating private or confidential data. However, without optimization, running state-of-the-art translation models locally often requires specialized hardware, such as one or more GPUs. Otherwise, the time to translate a single sentence can easily exceed one second on a standard CPU.

Higher decoding speeds can be achieved by model optimization. In their 2019 submission to the Workshop on Neural Generation and Translation (WNGT) Efficiency Shared Task, Kim et al. (2019) successfully employed knowledge distillation, quantization, shortlisting (Jean et al., 2015) and a simpler recurrent unit design to bring the throughput of a translation model up to 3,600 words per second on a CPU, with a modest drop in the translation quality. Following this work, Bogoychev et al. (2020) reported further improvements with attention head pruning (Voita et al., 2019). Their work has been part of the Bergamot Research Project, which aims to bring offline translation models to a browser.[3]

Non-autoregressive (NAR) models present an alternative approach to model optimization, using different architecture and a different decoding algorithm which has lower time complexity. In NMT, a non-autoregressive decoding algorithm does not access previously decoded outputs, imposing conditional independence assumption on the output token probability distributions. This assumption allows for parallelization of the decoding, which can significantly reduce the latency of the translation system. On the other hand, it also presents a challenge to the language model, which usually leads to poorer translation quality.

[1] `https://microsoft.com/translator/`

[2] `https://translate.google.com/`

[3] `https://browser.mt/`

2. Non-Autoregressive Neural Machine Translation

The defining feature of a NAR model is the assumption of conditional independence between the output distributions across time steps. The output distribution in autoregressive models is defined as follows:

$$p(y|x) = \prod_{t=1}^{T_y} p(y_t|y_{<t}, x, \theta) \tag{2.1}$$

Unlike Equation 2.1, NAR models do not condition the output token probabilities on previously decoded outputs $y_{<t}$. The probability of an output sentence y given an input sequence x can then be modeled as:

$$p(y|x) = \prod_{t=1}^{T_y} p(y_t|x, \theta) \tag{2.2}$$

Although technically possible, making the outputs in RNN-based models conditionally independent does not reduce the time complexity because in RNNs, the value of each hidden state depends on the value of the preceding state. However, in the Transformer model, hidden states in each layer depend only on the states from the previous layer. This allows for parallel computation at the layer level

In the following paragraphs, we discuss the necessary alterations to the Transformer architecture. Since the outputs are conditionally independent, we cannot feed the previously decoded outputs into the Transformer decoder. We need to provide the input to the decoder and estimate the target length. The causal mask over decoder self-attention is now unnecessary.

Multimodality Problem. In one of the first applications of non-autoregressive models to NMT, Gu et al. (2018) describe the *multimodality problem* which arises when the outputs are conditionally independent.

When estimating the probability of a word on a given position, there may be multiple words which get a high probability. These words are the so-called *modes* of the distribution. In autoregressive models, once a word is selected, other modes are ignored in the following time steps. However, a non-autoregressive model does not base its decision for a given position on the preceding ones, so when multiple positions have multiple modes, the model has no means of coordinating the selection of modes across different time steps.

A well-known example of the multimodality problem is the translation of the sentence “thank you” into German, which has two equally likely translations: “vielen dank” and “danke schön.” In this case, the pair of German tokens “danke” and “vielen” create the two modes in the first position, and the tokens “dank” and “schön” are the modes in the second position. If an autoregressive model chooses to generate “danke” in the first position, the token “dank” in the second position will no longer receive high probability from the model. However, when a non-autoregressive model assigns high probabilities to the correct translations, it also has to assign high probabilities to the other (incorrect) two combinations, “danke dank” and “vielen schön” (Gu et al., 2018).

Decoder Inputs. A NAR Transformer decoder cannot receive the previously decoded tokens on the input. A solution proposed by Gu et al. (2018) is to use a simple fertility model, which also serves as the explicit target length estimator.

Compared to the autoregressive Transformer, the model has the following modifications. First, the inputs to the decoder are made up of the sequence of encoder inputs, either uniformly stretched to the predicted target sentence length, or copied using a fertility model. Second, the decoder self-attention does not use the causal mask, since all states can now attend to all other states in both directions. Third, a *positional attention* sub-layer is added to every decoder layer, where the positional encoding is used as queries and keys, and the decoder states as values. Gu et al. (2018) argue that providing positional information directly to the decoder layers could improve the potential of the decoder to model local reodering.

Knowledge Distillation. To tackle the multimodality problem from another angle, Gu et al. (2018) propose to use sequence-level knowledge distillation to create artificial training data (Kim and Rush, 2016). Knowledge distillation is based on training an autoregressive (AR) teacher model and using it to create artificial training data for a student model. The main idea is that the outputs of a teacher model will have a simpler structure than natural language, limiting the number of distribution modes.

According to an ablation study published by Gu and Kong (2021), using knowledge distillation is a crucial element in training non-autoregressive translation models, regardless the actual method used. Zhou et al. (2020) further study the effects of knowledge distillation strategies on the translation quality of NAR models. They link the data complexity to the potential of NAR modeling and find that knowledge distillation reduces the complexity, which in turn leads to improved model performance.

2.1 Limitations of Related Work

In this section, we take a high-level view of related approaches. We point out a few aspects that most of the literature has in common, as well as some issues regarding the evaluation and comparison to meaningful baselines.

Translation Quality. Weak autoregressive baseline models are a common element in the non-autoregressive literature. In most cases, the base variant of the Transformer model is used as baseline (and most of the base hyperparameters are used for the NAR model as well), arguing that having similar numbers of parameters makes the baseline and the NAR model comparable. A comparable model size is a valid point; on the other hand, it raises doubts about the scalability of the proposed approaches; the difference between large AR and NAR model variants might not be proportional.

Moreover, even the Transformer base model can be trained in a better way than as reported by Vaswani et al. (2017) and as referenced by many NAR papers as their baseline, achieving a better translation quality (Popel and Bojar, 2018). A reasonable improvement of the baseline could be achieved by training a student AR model on the knowledge-distilled data.

Decoding Speed. Similarly to the previous paragraph, the baselines that appear in the literature are usually weak also in terms of decoding speed. For example, Gu et al. (2018) report a latency of 408 ms for their AR model (a Transformer base), measured on a single Nvidia P100 GPU, without batching. In contrast, our implementation achieves a latency of around 100 ms with the same model under the same hardware and hyperparameter settings and no optimizations. Additionally, some articles cite a baseline AR latency of 607 ms (Wang et al., 2019; Guo et al., 2020), which is the result of Gu et al. (2018) with beam search, even though beam search is not used in the presented NAR models for the most part.

Evaluation Methodology. Another aspect to be addressed is the evaluation methodology itself. Setting aside the fact that automatic quality evaluation using BLEU is the only reported metric, perhaps complemented by a cursory manual evaluation on a small sample, the evaluation of the speed improvements is inconsistent and sometimes the interpretation of the results is outright wrong.

The main problem is that the actual decoding speed depends on a lot of factors that are not easily reproducible. The most obvious factor is perhaps the hardware on which the decoding speed is measured, followed by the implementation. The average decoding time per sentence is also heavily influenced by the batch size in batched decoding. Table 2.1 shows that these conditions vary wildly within the literature.

Publication	GPU type	CPU?	Batch
Gu et al. (2018)	P100	✗	1
Lee et al. (2018)	P100 or P40	✓	1
Kaiser et al. (2018)	GTX 1080	✗	1, 64
Ghazvininejad et al. (2019)	Possibly V100	✗	10
Sun et al. (2019)	P100	✗	1
Wang et al. (2019)	P100	✗	1
Li et al. (2019)	Possibly M40	✗	1
Ma et al. (2019)	TITAN X	✗	*various*
Ghazvininejad et al. (2020)	*Not reported*	✗	*unknown*
Shao et al. (2020)	TITAN X	✗	1
Guo et al. (2020)	GTX 1080 Ti	✗	1
Kasai et al. (2020)	V100	✗	1
Qian et al. (2021)	GTX 1080 Ti	✗	1?
Ran et al. (2021)	P40	✗	1
Gu and Kong (2021)	V100	✓	1
Du et al. (2021)	*Not reported*	✗	1
Huang et al. (2021)	V100	✗	1

Table 2.1: The hardware setting and decoding batch size for measuring the decoding speed as reported in a sample of papers described in this section.

A popular solution that takes the varying conditions into account for evaluating decoding speed is to report the relative speed-up measured between experiments within a single study. However, comparing these relative speed-up ratios between different papers disregards the actual decoding times – it is easier to achieve 20 times speed-up over a slow baseline than over a baseline which is faster.

Due to these reasons, we believe that we cannot show a fair comparison of the methods presented here on a graph showing the pareto frontier between the decoding speed and translation quality.

However, it is challenging to find a way to compare the contributions of different research groups objectively. One such effort is made by the organizers of the Efficiency Shared Task, now organized yearly at the Conference on Machine Translation (WMT; Heafield et al., 2020, 2021).

In the Efficiency Shared Task, submissions are evaluated both in terms of translation quality and decoding speed under various settings. All submissions are evaluated on the same hardware, hosted by Amazon Web Services. To amortize the effects of data and model loading, the decoding time is measured on a dataset that contains one million sentences. The speed is measured in five different scenarios: GPU latency (decoding without batching) and throughput (decoding with the optimal batch size), latency and throughput on a single CPU, and throughput on a multi-core CPU. The main findings of the shared task are that autoregressive models can be optimized to achieve latency of 5–20 ms even with inexpensive hardware, without sacrificing much in terms of translation quality (Heafield et al., 2021).

Our final observation is that measuring latency on a single GPU without batching is a setup that favors NAR models, even though other scenarios should also be considered for real-world applications. In batched GPU decoding, the differences between AR and NAR models fade, because batch-parallelization of AR models makes up for time-parallelization of NAR models. On the other hand, in an online decoding scenario on a CPU, only limited parallelization is possible, so the advantage of NAR models is also diminished. We present detailed analysis of this behavior in Chapter 4.

3. Connectionist Temporal Classification

This chapter is based on the paper "End-to-End Non-Autoregressive Neural Machine Translation with Connectionist Temporal Classification", *joint work with Jindřich Libovický, published at EMNLP 2018.*

In this chapter, we lay the foundations for the NAR approaches to NMT studied in this thesis. We describe our experiments with an architecture based on the connectionist temporal classification (CTC) loss (Libovický and Helcl, 2018).

CTC (Graves et al., 2006) is a method for training neural networks on sequential data. Originally applied to the phonetic labelling task, but later successfully adapted in related areas, including automatic speech recognition (ASR) or handwriting recognition (Liwicki et al., 2007; Eyben et al., 2009; Graves and Jaitly, 2014).

The main strength of CTC becomes evident in tasks where the input and output labels are weakly or not at all aligned, for example, in situations where the observed input sequence is considerably longer than the target output sequence – hence the application to ASR, where the number of extracted features per second is higher than the number of phonemes uttered per second.

Training neural networks with CTC is independent of the actual neural network architecture. The CTC loss function can be applied on any network with sequential outputs. Thus, this method is applicable to both recurrent neural networks (RNNs) and the Transformer model.

Models trained with the CTC assume that the alignment between the input states (e.g. a group of frames in an audio signal) and the output states (e.g. a phoneme) is unknown. A variable number of frames in a row can encode a single phoneme. Similarly, in translation, multiple words in the source language may correspond to any number of (even non-consecutive) words in the target language.

The idea behind CTC is to allow some states to produce no output. This is achieved by introducing a special blank token into the target vocabulary. Optionally, identical outputs produced by multiple consecutive states may be merged and considered a single output. Because of these properties, there are groups of equivalent output sequences, which all represent the same target, as illustrated in Figure 3.1.

$$\text{a cat sat on a mat} = \begin{cases} \text{a <blank> cat sat on a <blank> mat} \\ \text{a a cat cat sat on a mat} \\ \text{a <blank> cat cat sat on a mat} \\ \dots \end{cases}$$

Figure 3.1: A group of output sequences of equal length which all represent the same target in CTC.

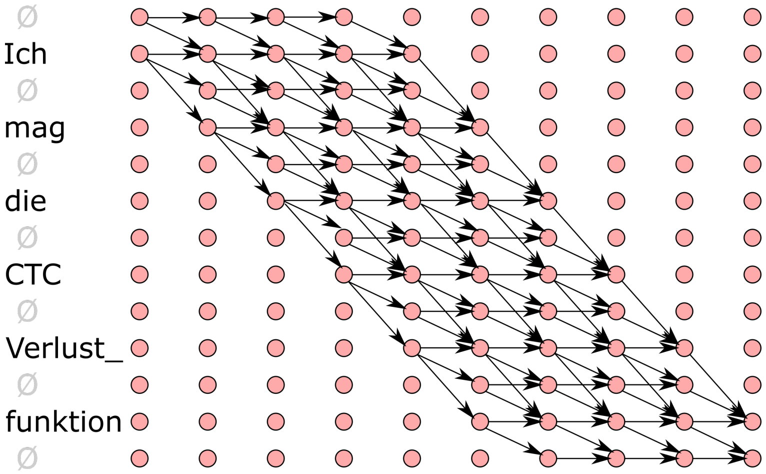


Figure 3.2: An illustration of the algorihm for the CTC loss computation. Each node denotes producing either a token from the label sequence, or the blank token. Each path from one of the two top-left nodes to one of the two bottom-right nodes corresponds to one of the equivalent sequences.

In standard sequence-to-sequence architectures, the value of the loss function is defined as the sum of cross entropies of the output distributions with respect to the target sequence.In CTC, the loss is defined as the sum of cross-entropy losses of all output sequences equivalent to the given target sequence:

$$\mathcal{L}_{\text{CTC}}(\theta) = - \sum_{(x,y)\in D} \sum_{y' \sim y} \log p(y'|x,\theta) \tag{3.1}$$

where $\sim$ denotes the equivalence relation.

The inner summation in Equation 3.1 is computed over all possible sequences equivalent to the label sequence. For technical purposes, the label sequences are limited to a fixed length, which greatly reduces the number of acceptable hypotheses. However, the number of equivalent hypotheses of a given length still grows exponentially with the sequence length – in CTC, the fixed length is always set to be longer than the label sequence.

The summation over the large set of equivalent sequences can be implemented using dynamic programming. When both the length of the output and the length of the target sequences are known, there is a constant number of blank tokens to be generated. The process of computing the loss of the whole output sequence is divided into computing the partial losses with respect to the possible label prefixes.

The CTC loss computation is illustrated in Figure 3.2. The rows represent tokens from the label sequence plus the optional blank tokens. The columns represent the output sequence. Each node in the graph denotes generating a label from an output distribution. The arrows show valid transitions between the generation steps. An arrow can only go down one or two rows, or horizontally. The horizontal arrows denote repeated generation of the same label. These labels are later merged to form a single output. An arrow can only go two rows down when the skipped row corresponds to the blank token, so no target tokens are left out. Therefore, each path in the diagram shows one of the equivalent sequences that lead to generating the given label sequence.

Using the idea that many of the paths from left to right in the diagram share segments, we can apply dynamic programming to compute the sum of losses across all paths without the need to enumerate each of them. A node on coordinates (i, j) stores the accumulated losses for all path prefixes that lead to the node, summed with the negative log-likelihood of the label on the i-th row being generated by the j-th output state. The two bottom-right nodes then store the sum of losses of all the paths.

Since the CTC loss function is differentiable with respect to the model parameters, the training of the network can be done in the standard way using the backpropagation algorithm.

3.1 Model Architecture

As mentioned in the previous section, training models with CTC does not impose any requirements on the model architecture. In our experiments, we aim to make a reasonable comparison between our proposed approach and the state-of-the-art autoregressive models. We adapt the Transformer model and use similar hyperparameters where applicable.

Non-autoregressive models generate the outputs in parallel, which requires that the output length is known beforehand. In autoregressive models, the end of the sequence is indicated by a special end symbol, and the constraint on maximum length is merely a technical aspect.

To maximize the ability to output empty tokens to the full extent, the output length should be set to a higher number than the length of the target sequence. Since the length estimation does not need to be accurate, we select a number k, and we set the target sequence length to be k times longer than the source length. Note that in case the selected length is shorter than the label sequence, the model will not be able to generate the whole target sequence.

We implement the source-to-target length expansion by linear projections and state splitting. This mechanism is illustrated in Figure 3.3. After a given Transformer layer completes its computation, we linearly project the states $h_1, \ldots, h_{T_x} \in \mathbb{R}^d$ into $\mathbb{R}^{kd}$. Then, we split each of these projections into k parts, which results in a k-times longer

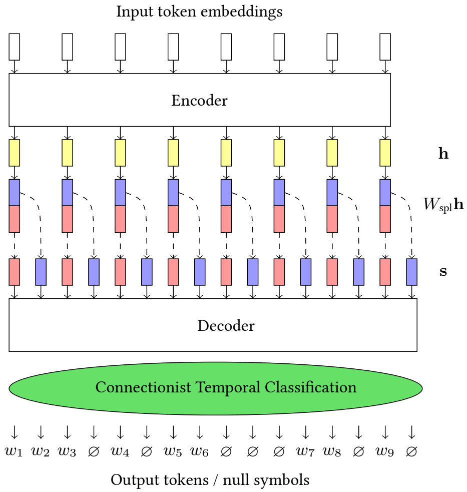

Input token embeddings
Encoder
h
$W_{\text{spl}}\mathbf{h}$
s
Decoder
Connectionist Temporal Classification
w_1 w_2 w_3 $\varnothing$ w_4 $\varnothing$ w_5 w_6 $\varnothing$ $\varnothing$ $\varnothing$ w_7 w_8 $\varnothing$ w_9 $\varnothing$
Output tokens / null symbols

Figure 3.3: The scheme of the non-autoregressive architecture with state splitting and CTC. Image source: Libovický and Helcl (2018), Figure 1.

sequence of states $s_1, \ldots, s_{k \cdot T_x}$ of the original dimension d:

$$s_{ck+b} = (W_{\text{spl}} h_c + b_{\text{spl}})_{bd:(b+1)d} \tag{3.2}$$

for $b = 0 \ldots k-1$ and $c = 1 \ldots T_x$ where $W_{\text{spl}} \in \mathbb{R}^{d \times kd}$ and $b_{\text{spl}} \in \mathbb{R}^{kd}$ are the trainable projection parameters.

We experiment with two placement options for the state-splitting layer. First, we try placing the state splitting at the end of the Transformer layer stack. In this scenario, there are 12 Transformer encoder layers, followed by the state-splitting layer, whose outputs are used in the output projection. In Table 3.1, this variant is called "Deep encoder". Second, we place the state-splitting layer in the middle of the Transformer layer stack, mimicking the 6-layer encoder-decoder architecture of the AR Transformer model. We use the name "Encoder-decoder" for this variant in Table 3.1. In the second variant, cross-attention can be included in the second half of the layers, which attends to the states right after state splitting. We call this model "Encoder-decoder with pos. enc." in the results table.

3.2 Preliminary Experiments

We conducted experiments with the CTC-based non-autoregressive NMT (NAT) models described above on English–German and English–Romanian translation in both directions.

Data. In our experiments, we use the parallel data provided by the WMT organizers. For English–German, the training data consist of the Europarl corpus (Koehn, 2005), News commentary (Tiedemann, 2012), and Common Crawl.[1] For validation, we use the WMT 13 test set (Bojar et al., 2013), and we evaluate translation quality on the WMT 14 test set (Bojar et al., 2014). For English–Romanian, the data consist of the Europarl corpus and the SETIMES corpus distributed by OPUS (Tiedemann, 2012). We use the development and test set from the WMT 16 (Bojar et al., 2016).

We preprocess the data with scripts from the `mosesdecoder` repository,[2] a part of the Moses translation toolkit (Koehn et al., 2007). Namely, we normalize the punctuation in the data, then we use the tokenizer and a truecaser. We segment the data using the wordpiece algorithm, creating a vocabulary of approximately 41k wordpieces for English–German, and 38k wordpieces for English–Romanian (Wu et al., 2016).

Models and Training. We implement and train our models in the Neural Monkey toolkit (Helcl and Libovický, 2017; Helcl et al., 2018). Neural Monkey is a higher-level deep learning toolkit implemented in TensorFlow (Abadi et al., 2015), aimed at fast prototyping using simple-format configuration files. We train all models for 10 epochs and select the best-scoring model based on the validation BLEU score. Training of the En–De models on a single Nvidia GeForce GTX 1080 GPU took approximately 4 weeks. Since the En–Ro parallel data was much smaller, training of the En–Ro models took 4 days.

Results. Table 3.1 compares the quantitative results of our NAR models with the methods proposed by Gu et al. (2018) and Lee et al. (2018). We use Sacrebleu (Post, 2018) to compute the BLEU scores of the model outputs.

The first model of Gu et al. (2018) represents the NAT model with fine-tuning, which minimizes the KL divergence between the output distributions of the teacher and student models. In the second row, noisy parallel decoding (NPD) with 100 samples is used. Using NPD can slow down the decoding because an additional scoring step is needed using an AR model. In theory, if enough parallelism is available, this doubles the latency.

The approach of Lee et al. (2018) is iterative. We show the performance of this method with 1 iteration and with 10 iterations. Note that using more iterations slows the decoding speed of the iterative model.

[1] `https://commoncrawl.org/`

[2] `https://github.com/moses-smt/mosesdecoder/`

	WMT 16		WMT 14	
	En → Ro	Ro → En	En → De	De → En
Gu et al. (2018)				
Autoregressive baseline	31.35	31.03	22.71	26.39
NAT + FT	27.29	29.06	17.69	21.47
NAT + FT + NPD (100 s)	29.79	31.44	19.17	23.20
Lee et al. (2018)				
Autoregressive baseline	31.93	31.55	23.77	28.15
1 iteration	24.45	25.73	13.91	16.77
10 iterations	29.32	30.19	21.61	25.48
Libovický and Helcl (2018)				
Autoregressive baseline	21.19	29.64	22.94	28.58
Deep encoder	17.33	22.85	12.21	12.53
+ weight averaging	18.47	24.68	14.65	16.72
+ beam search	18.70	25.28	15.19	17.58
Encoder-decoder	18.51	22.37	13.29	17.98
+ weight averaging	19.54	24.67	16.56	18.64
+ beam search	19.81	25.21	17.09	18.80
Encoder-decoder with pos. enc.	18.13	22.75	12.51	11.35
+ weight averaging	19.31	24.21	17.37	18.07
+ beam search	19.93	24.71	17.68	19.80

Table 3.1: Automatic evaluation of our CTC-based approach, compared to the two of the first non-autoregressive methods, along with autoregressive greedy-decoding baseline scores.

	CPU	GPU
AR, batch=1	2,247 ms	1,129 ms
AR, batch=100		127 ms
NAR, batch=1	397 ms	353 ms
NAR, batch=100		41 ms

Table 3.2: The average times to decode one sentence under different conditions.

The performance of the English–German CTC-based models is similar to single-pass models. However, the performance of the English–Romanian models is poor. This is probably due to issues with Romanian diacritics in the data, as suggested also by the poorer performance of our autoregressive baseline. Also, the enhanced techniques (NPD and iterative refiniment) outperform our method in all scenarios, but at a computational cost.

We also evaluate the effect of weight averaging and beam search. As opposed to decoding from an AR model, beam search in CTC-based NAR models operates only on the output distributions from the single decoding step. Although in NAR models we are able to find the most likely sequence by taking the argmax of the output distribution in each position, this way we only find a single alignment. Using beam search takes into account more alignments, and the sum of the alignment probabilities can be higher than the probability of the sentence generated by argmax decoding. We see from Table 3.1 that especially weight averaging brings a substantial improvement of the translation quality.

The decoding speed for AR and NAR models under different conditions is shown in Table 3.2. We report the average time to decode a single sentence on the WMT 15 test set (Bojar et al., 2015), which consists of 2,169 sentence pairs. In the CPU setting, we are using a CPU machine with the TensorFlow session configured to use 12 CPU threads. GPU results are measured on a single Nvidia GeForce GTX 1080 GPU. We show the average latency as well as the average time to decode a sentence with batch decoding.

Figure 3.4 shows the decoding latencies on sentences from the WMT 15 test set (Bojar et al., 2015). As we can see from the relationship between the source sentence length and the decoding time, the NAR model exhibits a lower time complexity than the AR model. The latencies in the plot were measured on a CPU server using 12 threads. We use greedy decoding with a single sentence in a batch.

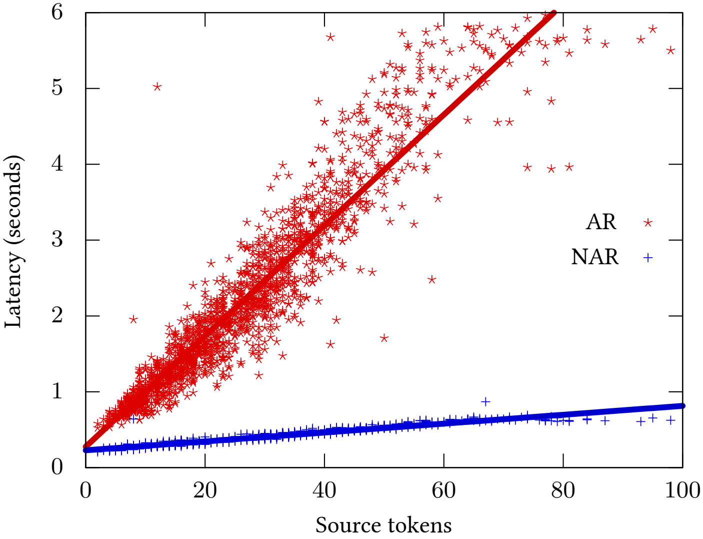


Figure 3.4: The relationship between the latency and the number of source tokens using an autoregressive and a non-autoregressive model

4. Experiments

The flaws in the evaluation methodology adopted by current research of NAT models make it difficult to identify approaches suitable in production-ready conditions. In this chapter, we present further experiments with NAT models trained with CTC. We focus on a fair comparison with other non-autoregressive approaches as well as state-of-the-art optimized autoregressive methods. In line with our previous experiments and also with the rest of the related work, we conduct experiments on English-German translation.

4.1 Autoregressive Teacher Models

> *This section is based on the paper* "The University of Edinburgh's English-German and English-Hausa Submissions to the WMT21 News Translation Task", *joint work with Pinzen Chen, Ulrich Germann, Laurie Burchell, Nikolay Bogoychev, Antonio Valerio Miceli Barone, Jonas Waldendorf, Alexandra Birch, and Kenneth Heafield, published at WMT 2021.*

In our English-German experiments, we use autoregressive models from our submission to the WMT 2021 News Translation Shared Task (Chen et al., 2021). We use these models both as strong AR baselines and as teacher models for generating distilled data for the NAR student models, described in Section 4.2. The models were trained on cleaned parallel data augmented with backtranslated monolingual data, using Marian, a C++ toolkit for training and decoding from NMT models (Junczys-Dowmunt et al., 2018).

Data Cleaning. We prepare the training dataset consisting of the following parts: First, we use clean parallel data from the Europarl corpus (Koehn, 2005), the Tilde MODEL – RAPID corpus (Rozis and Skadiņš, 2017), and the News Commentary corpus from OPUS (Tiedemann, 2012). Next, we include sources of crawled parallel data from the web, which are considered noisy. These include Paracrawl (Esplà et al., 2019), Common Crawl[1], WikiMatrix (Schwenk et al., 2019), and the Wikipedia Parallel Titles Corpus[2]. Finally, we use backtranslation (Sennrich et al., 2016) of monolingual data obtained from News Crawl. We train our own models to generate the backtranslations on cleaned parallel data, as described below.

We perform the following filtering techniques on the gathered data (both clean and noisy) to improve the overall quality of the parallel data.

[1] `https://commoncrawl.org/`

[2] `https://linguatools.org/tools/corpora/wikipedia-parallel-titles-corpora/`

We start with deterministic rule-based filtering[3] and deduplication. We remove all sentence pairs containing non-printing characters, empty sentences, and sentences longer than 120 words; we also remove all sentence pairs with length ratio of less than 0.6 (0.4 for Wikititles), sentences in which over 40% of characters do not constitute tokens, and sentences in which more than 50% were non-alphabetic characters. We run language identification using fastText (Joulin et al., 2017, 2016) and remove all sentence pairs classified as not English-German.

To further clean the data, we apply dual cross-entropy filtering, as proposed by Junczys-Dowmunt (2018). We train two Transformer base models (one for each translation direction) for dual cross-entropy filtering using the clean part of the data after the rule-based cleaning step.

We score the crawled part of the parallel data using the trained models and we sort the sentence pairs according to the score. To estimate how many of the crawled sentence pairs can we consider clean, we train the big variant of the Transformer translation models in both directions on different amounts of the scored data (following the teacher model hyperparameter settings in Table 4.1). We use 25%, 50%, 75% and 100% of the data, taking the highest-scoring sentence pairs for training. Based on the BLEU scores achieved by the Transformer big models on the development data (the WMT 2019 test set, Barrault et al., 2019), we declare the 75% as the "clean" portion of the crawled data.

Backtranslation. We train three additional translation models (four in total with the one from the data cleaning step) on the filtered parallel dataset to create backtranslations (Sennrich et al., 2016). We use the same hyperparameter settings as for the teacher models (see Table 4.1), except for different random seeds for parameter initialization.

We translate the monolingual data using the four models in an ensemble. We limit the target sentence length to 150 tokens and use beam search decoding with 6 hypotheses in the beam, with the length normalization parameter set to 1.

As the source of the monolingual data, we use the News Crawl datasets from years 2018, 2019, and 2020, as released by the WMT organizers (Bojar et al., 2018; Barrault et al., 2019, 2020). In total, we gathered 91 million English sentences for backtranslation into German and 146 million German sentences for backtranslation into English.

We follow the approach of Caswell et al. (2019) and we tag the backtranslated sentences with a special token on a first position.

Teacher Model Training. We train the teacher models on shuffled concatenation of the authentic parallel and tagged backtranslated data. As with the backtranslation models, we train four models with different seeds for random initialization in each direction. We show the hyperparameter values in Table 4.1.

[3] `https://github.com/browsermt/students/blob/master/train-student/clean/`

Parameter	Marian config variable	Value
No. of encoder layers	`enc-depth`	6
No. of decoder layers	`dec-depth`	6
Model dimension	`dim-emb`	1,024
Feed-forward state dimension	`transformer-dim-ffn`	4,096
Attention heads	`transformer-heads`	16
Vocabulary size		32,000
Optimizer method	`optimizer`	adam
β_1		0.9
β_2	`optimizer-params`	0.998
ϵ		10^{-9}
No. of batches per update	`optimizer-delay`	2
Fit batch to available memory	`mini-batch-fit`	true
Learning rate	`learn-rate`	10^{-4}
Learning rate warmup	`lr-warmup`	8,000
Learning rate decay	`lr-decay-inv-sqrt`	8,000
Gradient clipping[4]	`clip-norm`	0

Table 4.1: The hyperparameters of the teacher models. The same values were used for training the models for backtranslation.

After the training on mixed parallel and backtranslated data converged, we continued the training of the models on parallel data only.

Knowledge Distillation. The teacher models are used to create artificial targets for the student models (Kim and Rush, 2016). For each translation direction, we translate the source side of the parallel data and the authentic monolingual data (in the source language) using the ensemble of the teacher models. We do not create wholly synthetic datasets by forward-translating backtranslated data.

4.2 Student Models

In this section, we describe our non-autoregressive student models. We implemented the CTC-based Transformer architecture in the Marian toolkit (Junczys-Dowmunt et al., 2018). For the computation of the CTC loss, we use the warp-ctc library,[5] an efficient parallelized implementation for GPUs (Amodei et al., 2016).

[4] We did not use gradient clipping because of issues in the Marian toolkit implementation.

[5] `https://github.com/baidu-research/warp-ctc/`

Model	Base architecture	Encoder layers	Decoder layers
Teacher	Transformer big	6	6
Large	Transformer big	6	6
Base	Transformer base	6	6
Small		3	3
Micro		2	2
Tiny		1	1

Table 4.2: The hyperparameters of the student models.

We experiment with different hyperparameter settings that control the size of the student model. Our baseline non-autoregressive model is a big Transformer model with 6 encoder layers, followed by the state-splitting layer, and another 6 decoder layers. Apart from the CTC-specific configuration, we use the same hyperparameters as in the teacher models, shown in Table 4.1. In addition, we train four smaller models based on the Transformer base hyperparameters – using model dimension of 512, feed-forward state dimension of 2,048, and 8 attention heads. Each smaller model uses a different number of encoder and decoder layers. We show the number of encoder and decoder layers in Table 4.2. In all settings, we use the splitting factor of 3 in the state splitting layer.

4.3 Results

In this section we analyze the results both in terms of translation quality in Section 4.3.1 and decoding speed in Section 4.3.2. We discuss the results from the perspectives of the related literature on the NAT models and the submissions to the Efficiency Shared Task in Section 4.3.3.

4.3.1 Translation Quality

We compare the results of our models to both related work on NAT and to the results of the WMT 21 Efficiency Shared Task which features highly optimized AR translation models (Heafield et al., 2021). In the NAR literature, the WMT 14 test set (Bojar et al., 2014) is used as a standard benchmark. On the other hand, the efficiency task uses the recent test set from WMT 21 (Akhbardeh et al., 2021). In this section, we present the results on both datasets.

Results on the WMT 21 test set. In line with the evaluation methodology of the efficiency task, we also measure COMET (Rei et al., 2020) and ChrF score (Popović, 2015). We do not perform human evaluation.

Table 4.3 shows the results of automatic evaluation on the WMT 21 news translation test set. In the English → German direction, the test set consists of 1,002 sentences along with three different reference translations. In the German → English direction, there are 1,000 sentences with two reference translations each. We measure multi-reference BLEU score using Sacrebleu (Post, 2018) and we report confidence intervals computed with bootstrap resampling. We report the Sacrebleu signatures for English → German[6] and German → English[7] directions. We compute the ChrF score separately on each reference translation set using Sacrebleu, and we report the average.[8] Finally, we compute the COMET scores with the `wmt20-comet-da` model of the COMET version `dd2298` (1.0.0rc9).

We observe that in both translation directions, the AR models outperform the NAR models. The performance gap between the models grows further with beam search and ensembling. We can also see that knowledge distillation has a positive effect on both AR and NAR models, with the student AR model matching the performance of the ensemble of four large teacher models. We also note that the difference in the COMET score is bigger than in BLEU, which might suggest that NAR models would rank poorly in human evaluation, despite achieving reasonable BLEU scores.

The results on the test set confirm the ranking of the NAR models as seen during training, including the interesting exception of the English → German Micro model. Otherwise, the larger the student model is, the better scores it achieves.

Finally, we see that using a lexical shortlist does not have an effect on the translation quality in all model variants. However, we see that shortlisting impairs the performance of the Transformer base model when decoding with beam search.

Results on the WMT 14 test set. We present automatic evaluation results measured on the WMT 14 test set. Since many of the related approaches stop the training early after 300 thousand updates (Gu et al., 2018; Gu and Kong, 2021), we report the scores of our models both at this point in training, and after the training reached convergence.

Table 4.4 shows the WMT 14 BLEU scores with checkpoint averaging. In each variant, we take the average parameters of the five best scoring models as measured on the validation set (either before the 300,000th update or overall).

4.3.2 Decoding Time

In this section, we analyze the decoding speed of the English → German NAR models. We aim at recreating the evaluation conditions following the WMT 21 Efficiency Shared Task (Heafield et al., 2021). We measure the decoding latency and throughput in different hardware environments.

[6] En→De: `nrefs:3|bs:1000|seed:12345|case:mixed|eff:no|tok:13a|smooth:exp|version:2.0.0`

[7] De→En: `nrefs:2|bs:1000|seed:12345|case:mixed|eff:no|tok:13a|smooth:exp|version:2.0.0`

[8] ChrF; En↔De: `nrefs:1|case:mixed|eff:yes|nc:6|nw:0|space:no|version:2.0.0`

En → De	Full output projection			Shortlist		
	ChrF	COMET	BLEU	ChrF	COMET	BLEU
Single greedy AR						
Large	59.2	0.4110	50.5 ±1.3	59.2	0.4124	50.6 ±1.3
Base	58.1	0.3881	47.9 ±1.3	58.1	0.3875	47.9 ±1.2
Single beam AR						
Large	58.8	0.4053	50.8 ±1.3	58.8	0.4144	47.9 ±1.2
Base	57.9	0.3873	48.0 ±1.3	55.1	0.2666	39.3 ±1.1
Ensemble beam AR						
Large	59.5	0.4332	52.2 ±1.3	59.4	0.4303	52.2 ±1.3
Student AR						
Base	59.5	0.4550	51.6 ±1.2	59.6	0.4564	51.6 ±1.2
NAR models						
Large	58.6	0.1485	47.8 ±1.2	58.7	0.1442	47.7 ±1.2
Base	56.3	-0.0521	41.8 ±1.1	56.3	-0.0545	41.8 ±1.1
Small	56.2	-0.0752	41.9 ±1.1	56.2	-0.0773	41.9 ±1.2
Micro	57.3	-0.0083	43.5 ±1.1	57.4	-0.0085	43.6 ±1.1
Tiny	53.6	-0.3333	34.7 ±1.0	53.8	-0.3346	34.8 ±1.0

De → En	Full output projection			Shortlist		
	ChrF	COMET	BLEU	ChrF	COMET	BLEU
Single greedy AR						
Large	61.9	0.5868	48.4 ±1.3	61.9	0.5866	48.5 ±1.3
Base	61.0	0.5532	47.0 ±1.3	60.5	0.5534	47.1 ±1.3
Single beam AR						
Large	61.5	0.5885	49.2 ±1.2	61.2	0.5659	43.9 ±1.1
Base	60.7	0.5534	47.4 ±1.3	58.0	0.4591	38.5 ±1.2
Ensemble beam AR						
Large	62.0	0.5954	50.6 ±1.3	62.3	0.5970	50.8 ±1.3
Student AR						
Base	63.3	0.6115	51.1 ±1.3	63.3	0.6112	51.1 ±1.3
NAR models						
Large	61.6	0.3296	46.4 ±1.4	61.6	0.3288	46.4 ±1.4
Base	61.4	0.2957	45.8 ±1.3	61.4	0.2663	45.7 ±1.3
Small	61.0	0.2462	44.6 ±1.3	61.0	0.2454	44.6 ±1.3
Micro	59.6	0.1475	42.3 ±1.4	59.6	0.1468	42.3 ±1.4
Tiny	55.9	-0.1558	34.4 ±1.3	55.9	-0.1560	34.4 ±1.3

Table 4.3: Quantitative results of the German ↔ English translation models on the WMT 21 test set using ChrF, COMET, and BLEU.

	En → De		De → En	
Model	300k	Final	300k	Final
Large	27.7	28.4	30.0	31.3
Base	22.4	23.7	28.1	30.3
Small	22.5	23.6	26.7	29.1
Micro	23.7	25.0	25.1	27.5
Tiny	19.0	20.3	19.6	21.7

Table 4.4: The BLEU scores of the *averaged* models on the WMT 14 test set after 300k updates and at the end of the training.

The decoding time is measured on a dataset containing one million sentences to minimize the effect of the model loading overhead.

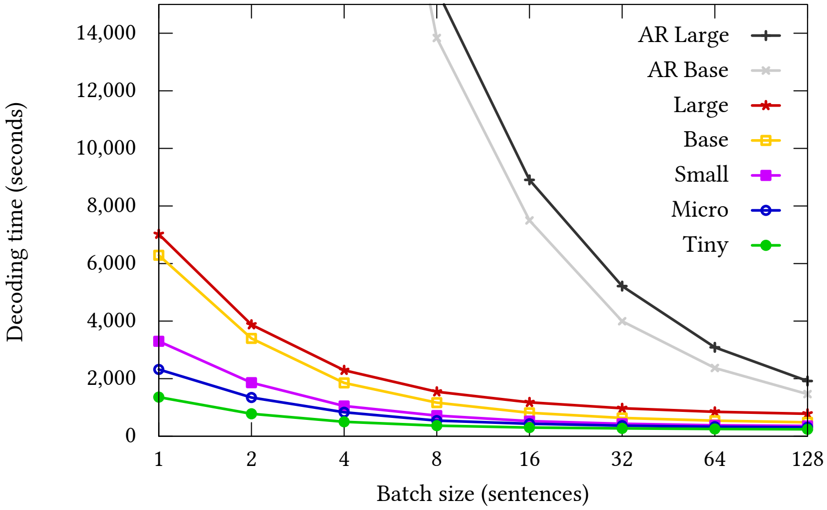
For measuring the CPU times, we use the same environment as the WMT 21 shared task organizers, which is a dual-socket Intel Xeon Gold 6354 from Oracle Cloud, a 36-core CPU server. For GPU efficiency, we use an Nvidia A100 GPU.

GPU Latency and Throughput. Figure 4.1 shows the relationship between the batch size and the decoding time in seconds on the A100 GPU. For AR models, increasing the batch size heavily reduces the overall decoding speed and eventually surpasses the large NAR models. Increasing the batch size speeds up the decoding in NAR models as well, but the effect is diminished for larger batch sizes.

From this point of view, the optimal scenario for NAR models is a situation in which the system runs in online mode, i.e. with a batch size of 1, or with a batch size of a few sentences. On a faster GPU with more threads, AR models need a larger batch size to meet the speed of NAR models.

CPU Latency and Throughput. In Figure 4.2, we show the CPU decoding times of the trained models using 36 CPU cores with different batching settings. We see similar trends to GPU decoding – the NAR models are faster with smaller batch sizes. The AR models eventually match the decoding speed of the NAR models as the batch size increases. However, there is a considerable difference between the large and base models in both AR and NAR variants.

We also notice that increasing the batch size can slow the decoding speed down. There may be several reasons for this behavior. First, the size of the shortlist grows proportionally with the batch size, as there are more possible target words. Second, when the batch is too large, much of the computation is wasted on the padded positions in shorter sentences. These issues are not evident in GPU decoding due to a higher level of parallelism.



Batch size	1	2	4	8	16	32	64	128
AR – Large		53,902	29,369	15,351	8,907	5,216	3,090	1,918
AR – Base		47,145	25,745	13,836	7,498	3,997	2,371	1,465
Large	7,020	3,874	2,292	1,547	1,179	973	850	782
Base	6,289	3,400	1,854	1,166	816	635	542	485
Small	3,300	1,860	1,051	717	526	434	380	357
Micro	2,322	1,345	833	544	433	367	332	311
Tiny	1,360	780	503	367	301	273	252	243

Figure 4.1: Wall times to translate one million sentences (in seconds) on a single Nvidia *Ampere* A100 GPU with different batch size settings.

	Translation quality			Decoding time (seconds)		
	ChrF	COMET	BLEU	GPU, b>1	GPU, b=1	CPU, b>1
Edinburgh base	61.5	0.527	55.3	140	16,851	500
AR – Large (teacher)	59.2	0.411	50.5	1,918	*> 24h*	9,090
AR – Base (student)	59.5	0.455	51.6	1,465	*> 24h*	2,587
NAR – Large	58.6	0.149	47.8	782	7,020	7,434
NAR – Micro	57.3	-0.008	43.5	311	2,322	897

Table 4.5: Comparison of our models with the Edinburgh "base" model submitted to the WMT Efficiency Shared Task (Behnke et al., 2021). Columns denoted *b>1* show the best result using batching, *b=1* is measured with a single sentence in the batch. CPU times were measured using 36 CPU cores.

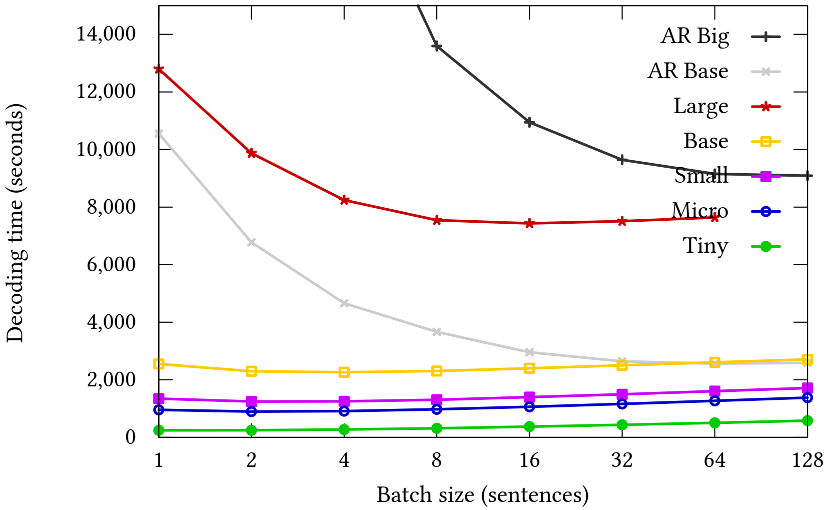
It is apparent from the data table in Figure 4.2 that the use of a lexical shortlist improves the decoding speed of both AR and NAR models. For clarity, we only plot the measured decoding times with shortlisting. We see that changing the batch size has a similar effect in both cases, perhaps with the exception of the micro and tiny models, which benefit greatly from the shortlist in combination with a small batch size.

In case of the NAR model with batch size of 128, we ran across an out-of-memory error due to a limitation in the Marian implementation. When using multiple CPUs for the decoding, the program makes a copy of the whole model for each CPU core. When the batch size is too large, this will eventually fill the whole RAM, and the process is killed.

4.3.3 Discussion

Table 4.4 in Section 4.3.1 shows that we achieve state-of-the-art scores in non-autoregressive translation on the WMT 14 test set, outperforming both single-step and iterative methods. However, the results on the WMT 21 test set in Table 4.3 show that there is still a great deal of room for improvement, especially when looking at the COMET scores. We believe that these findings should motivate future research not to evaluate translation quality exclusively on the WMT 14 test set.

We compare our models with one of the best performing submissions in the efficiency task – the University of Edinburgh's "base" model (Behnke et al., 2021) – in Table 4.5. It is clear from the comparison that the Edinburgh autoregressive model is superior to our NAR models in most regards. One exception is the GPU decoding latency. As we discuss in Section 2.1, these conditions are the only scenario considered in most of the related studies.



Batch size	1	2	4	8	16	32	64	128
Full output								
AR – Large	52,087	33,189	23,478	16,731	12,166	10,151	9,370	9,244
AR – Base	16,293	9,727	7,191	5,420	3,635	2,925	2,707	2,664
Large	14,542	11,320	9,534	8,657	8,357	8,247	8,238	
Base	3,508	3,126	2,979	2,927	2,921	2,920	2,934	2,948
Small	2,346	2,079	1,975	1,933	1,921	1,921	1,936	1,950
Micro	1,952	1,728	1,641	1,600	1,588	1,587	1,607	1,627
Tiny	784	719	697	687	685	694	701	719
Shortlist								
AR – Large	41,168	27,977	18,914	13,597	10,946	9,643	9,154	9,090
AR – Base	10,555	6,776	4,660	3,664	2,957	2,643	2,564	2,587
Large	12,799	9,870	8,245	7,545	7,434	7,511	7,639	
Base	2,549	2,298	2,263	2,306	2,399	2,503	2,609	2,707
Small	1,346	1,246	1,250	1,306	1,399	1,497	1,606	1,714
Micro	958	897	913	974	1,062	1,163	1,271	1,380
Tiny	244	246	273	316	373	437	506	582

Figure 4.2: Wall times to translate one million sentences (in seconds) on 36 CPU cores with different batch size settings. The graph shows the decoding times with shortlisting.

5. Conclusions

In this thesis, we explored the possibilities of NAR models for NMT. We described the commonly used NMT models and showed techniques to effectively train them and to use them for translation.

We presented a comprehensive survey of the related research efforts that have been made so far in this rapidly developing field. We described NAR methods based on latent variables, iterative refinement, or on the relaxed alignment between the output predictions and the ground-truth labels.

We pointed out some of the drawbacks in related NAR approaches, such as using weaker baselines both in terms of translation quality and decoding speed, and identified some of the flaws in the evaluation methodology.

In the first of the two experimental chapters (Chapter 3), we proposed a novel method for NAR NMT based on CTC which has been adopted by the NAR research community as one of the basic approaches since its original publication.

In Chapter 4, we aim at improving our CTC-based approach by using knowledge distillation using a strong AR teacher model. We compare our results to both the WMT 14 benchmark and the results of the WMT 21 Efficiency Shared Task (Heafield et al., 2021). We find that even though our models are among the best performing on WMT 14, there is a large room for improvement when we compare them to highly optimized autoregressive models submitted to the efficiency task.

To conclude, over the past few years, the research community has set its hopes on the NAR models because, in theory, the decoding has lower time complexity. Moreover, the literature on this topic often claims that NAR models already match the performance of their AR counterparts. However, our findings suggest that, in fair comparison with strong efficient baselines, many research studies concluding that NAR models are superior over AR models may be overclaiming. To avoid these problems in the future, research of NAR models should take into account findings of the efficiency task, evaluate the translation quality also on newer test sets, and measure the decoding speed under various conditions, such as batched GPU and CPU decoding.

Bibliography

Abadi, M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Available at: https://www.tensorflow.org/. Software available from tensorflow.org.

Akhbardeh, F. et al. Findings of the 2021 Conference on Machine Translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, p. 1–93, Online, November 2021. Association for Computational Linguistics. Available at: https://aclanthology.org/2021.wmt-1.1.

Amodei, D. – Ananthanarayanan, S. – Anubhai, R. – Bai, J. – Battenberg, E. – Case, C. – Casper, J. – Catanzaro, B. – Cheng, Q. – Chen, G. – others. Deep Speech 2: End-to-end Speech Recognition in English and Mandarin. In *International conference on machine learning*, p. 173–182. PMLR, 2016.

Barrault, L. – Bojar, O. – Costa-jussà, M. R. – Federmann, C. – Fishel, M. – Graham, Y. – Haddow, B. – Huck, M. – Koehn, P. – Malmasi, S. – Monz, C. – Müller, M. – Pal, S. – Post, M. – Zampieri, M. Findings of the 2019 Conference on Machine Translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, p. 1–61, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5301. Available at: https://aclanthology.org/W19-5301.

Barrault, L. et al. Findings of the 2020 Conference on Machine Translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, p. 1–55, Online, November 2020. Association for Computational Linguistics. Available at: https://aclanthology.org/2020.wmt-1.1.

Behnke, M. – Bogoychev, N. – Aji, A. F. – Heafield, K. – Nail, G. – Zhu, Q. – Tchistiakova, S. – Linde, J. – Chen, P. – Kashyap, S. – Grundkiewicz, R. Efficient Machine Translation with Model Pruning and Quantization. In *Proceedings of the Conference on Machine Translation at the 2021 Conference on Empirical Methods in Natural Language Processing*, Punta Cana, Dominican Republic, November 2021. Available at: https://kheafield.com/papers/edinburgh/wmt21-speed.pdf.

Bogoychev, N. – Grundkiewicz, R. – Aji, A. F. – Behnke, M. – Heafield, K. – Kashyap, S. – Farsarakis, E.-I. – Chudyk, M. Edinburgh's Submissions to the 2020 Machine Translation Efficiency Task. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, p. 218–224, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.ngt-1.26. Available at: https://aclanthology.org/2020.ngt-1.26.

Bojar, O. – Buck, C. – Callison-Burch, C. – Federmann, C. – Haddow, B. – Koehn, P. – Monz, C. – Post, M. – Soricut, R. – Specia, L. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, p. 1–44, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. Available at: https://aclanthology.org/W13-2201.

Bojar, O. – Buck, C. – Federmann, C. – Haddow, B. – Koehn, P. – Leveling, J. – Monz, C. – Pecina, P. – Post, M. – Saint-Amand, H. – Soricut, R. – Specia, L. – Tamchyna, A. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, p. 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3302. Available at: https://aclanthology.org/W14-3302.

Bojar, O. – Chatterjee, R. – Federmann, C. – Haddow, B. – Huck, M. – Hokamp, C. – Koehn, P. – Logacheva, V. – Monz, C. – Negri, M. – Post, M. – Scarton, C. – Specia, L. – Turchi, M. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, p. 1–46, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3001. Available at: https://aclanthology.org/W15-3001.

Bojar, O. et al. Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, p. 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2301. Available at: https://aclanthology.org/W16-2301.

Bojar, O. – Federmann, C. – Fishel, M. – Graham, Y. – Haddow, B. – Koehn, P. – Monz, C. Findings of the 2018 Conference on Machine Translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, p. 272–303, Belgium, Brussels, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6401. Available at: https://aclanthology.org/W18-6401.

Caswell, I. – Chelba, C. – Grangier, D. Tagged Back-Translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, p. 53–63, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5206. Available at: https://aclanthology.org/W19-5206.

Chen, P. – Helcl, J. – Germann, U. – Burchell, L. – Bogoychev, N. – Barone, A. V. M. – Waldendorf, J. – Birch, A. – Heafield, K. The University of Edinburgh's English-German and English-Hausa Submissions to the WMT21 News Translation Task. In *Proceedings of the Sixth Conference on Machine Translation*. Association for Computational Linguistics, 2021.

Du, C. – Tu, Z. – Jiang, J. Order-Agnostic Cross Entropy for Non-Autoregressive Machine Translation, 2021.

Esplà, M. – Forcada, M. – Ramírez-Sánchez, G. – Hoang, H. ParaCrawl: Web-scale parallel corpora for the languages of the EU. In *Proceedings of Machine Translation Summit XVII: Translator, Project and User Tracks*, p. 118–119, Dublin, Ireland, August 2019. European Association for Machine Translation. Available at: https://aclanthology.org/W19-6721.

Eyben, F. – Wöllmer, M. – Schuller, B. – Graves, A. From Speech to Letters – Using a Novel Neural Network Architecture for Grapheme Based ASR. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, p. 376–380. IEEE, 2009.

Ghazvininejad, M. – Levy, O. – Liu, Y. – Zettlemoyer, L. Mask-Predict: Parallel Decoding of Conditional Masked Language Models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, p. 6112–6121, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1633. Available at: https://aclanthology.org/D19-1633.

Ghazvininejad, M. – Karpukhin, V. – Zettlemoyer, L. – Levy, O. Aligned Cross Entropy for Non-Autoregressive Machine Translation. In III, H. D. – Singh, A. (Ed.) *Proceedings of the 37th International Conference on Machine Learning*, 119 / *Proceedings of Machine Learning Research*, p. 3515–3523. PMLR, 13–18 Jul 2020. Available at: http://proceedings.mlr.press/v119/ghazvininejad20a.html.

Graves, A. – Jaitly, N. Towards End-to-End Speech Recognition with Recurrent Neural Networks. In *International conference on machine learning*, p. 1764–1772. PMLR, 2014.

Graves, A. – Fernández, S. – Gomez, F. – Schmidhuber, J. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd International Conference on Machine Learning*, p. 369–376, Pittsburgh, PA, USA, June 2006. JMLR.org.

Gu, J. – Kong, X. Fully Non-autoregressive Neural Machine Translation: Tricks of the Trade. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, p. 120–133, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.11. Available at: `https://aclanthology.org/2021.findings-acl.11`.

Gu, J. – Bradbury, J. – Xiong, C. – Li, V. O. K. – Socher, R. Non-Autoregressive Neural Machine Translation. In *6th International Conference on Learning Representations, ICLR 2018*, Vancouver, BC, Canada, April 2018. Available at: `https://openreview.net/forum?id=B1l8BtlCb`.

Guo, J. – Xu, L. – Chen, E. Jointly Masked Sequence-to-Sequence Model for Non-Autoregressive Neural Machine Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 376–385, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.36. Available at: `https://aclanthology.org/2020.acl-main.36`.

Heafield, K. – Hayashi, H. – Oda, Y. – Konstas, I. – Finch, A. – Neubig, G. – Li, X. – Birch, A. Findings of the Fourth Workshop on Neural Generation and Translation. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, p. 1–9, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.ngt-1.1. Available at: `https://aclanthology.org/2020.ngt-1.1`.

Heafield, K. – Zhu, Q. – Grundkiewicz, R. Findings of the WMT 2021 Shared Task on Efficient Translation. In *Proceedings of the Conference on Machine Translation at the 2021 Conference on Empirical Methods in Natural Language Processing*, Punta Cana, Dominican Republic, November 2021. Available at: `https://kheafield.com/papers/edinburgh/wmt21-speedtask.pdf`.

Helcl, J. – Libovický, J. – Kocmi, T. – Musil, T. – Cífka, O. – Variš, D. – Bojar, O. Neural Monkey: The Current State and Beyond. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, p. 168–176, Boston, MA, March 2018. Association for Machine Translation in the Americas. Available at: `https://aclanthology.org/W18-1816`.

Helcl, J. – Libovický, J. Neural Monkey: An Open-source Tool for Sequence Learning. *The Prague Bulletin of Mathematical Linguistics*. Apr 2017, 107, 1, p. 5–17. ISSN 0032-6585.

Huang, C. – Zhou, H. – Zaïane, O. R. – Mou, L. – Li, L. Non-Autoregressive Translation with Layer-Wise Prediction and Deep Supervision, 2021.

Jean, S. – Cho, K. – Memisevic, R. – Bengio, Y. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, p. 1–10, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1001. Available at: `https://aclanthology.org/P15-1001`.

Joulin, A. – Grave, E. – Bojanowski, P. – Douze, M. – Jégou, H. – Mikolov, T. FastText.zip: Compressing Text Classification Models. *arXiv preprint arXiv:1612.03651*. 2016.

Joulin, A. – Grave, E. – Bojanowski, P. – Mikolov, T. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, p. 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics. Available at: `https://aclanthology.org/E17-2068`.

Junczys-Dowmunt, M. Dual Conditional Cross-Entropy Filtering of Noisy Parallel Corpora. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, p. 888–895, Belgium, Brussels, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6478. Available at: `https://aclanthology.org/W18-6478`.

Junczys-Dowmunt, M. – Grundkiewicz, R. – Dwojak, T. – Hoang, H. – Heafield, K. – Neckermann, T. – Seide, F. – Germann, U. – Aji, A. F. – Bogoychev, N. – Martins, A. F. T. – Birch, A. Marian: Fast Neural Machine Translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, p. 116–121, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-4020. Available at: `https://aclanthology.org/P18-4020`.

Kaiser, L. – Bengio, S. – Roy, A. – Vaswani, A. – Parmar, N. – Uszkoreit, J. – Shazeer, N. Fast Decoding in Sequence Models Using Discrete Latent Variables. In Dy, J. – Krause, A. (Ed.) *Proceedings of the 35th International Conference on Machine Learning*, 80 / *Proceedings of Machine Learning Research*, p. 2390–2399, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. Available at: `http://proceedings.mlr.press/v80/kaiser18a.html`.

Kasai, J. – Cross, J. – Ghazvininejad, M. – Gu, J. Non-Autoregressive Machine Translation with Disentangled Context Transformer. In III, H. D. – Singh, A. (Ed.) *Proceedings of the 37th International Conference on Machine Learning*, 119 / *Proceedings of Machine Learning Research*, p. 5144–5155. PMLR, 13–18 Jul 2020. Available at: `http://proceedings.mlr.press/v119/kasai20a.html`.

Kim, Y. – Rush, A. M. Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, p. 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. Available at: `https://aclanthology.org/D16-1139`.

Kim, Y. J. – Junczys-Dowmunt, M. – Hassan, H. – Fikri Aji, A. – Heafield, K. – Grundkiewicz, R. – Bogoychev, N. From Research to Production and Back: Ludicrously Fast Neural Machine Translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, p. 280–288, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5632. Available at: `https://aclanthology.org/D19-5632`.

Koehn, P. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of Machine Translation Summit X: Papers*, p. 79–86, Phuket, Thailand, September 13-15 2005. Available at: `https://aclanthology.org/2005.mtsummit-papers.11`.

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. – Herbst, E. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, p. 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. Available at: `https://aclanthology.org/P07-2045`.

Lee, J. – Mansimov, E. – Cho, K. Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 1173–1182, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1149. Available at: `https://aclanthology.org/D18-1149`.

Li, Z. – Lin, Z. – He, D. – Tian, F. – Qin, T. – Wang, L. – Liu, T.-Y. Hint-Based Training for Non-Autoregressive Machine Translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, p. 5708–5713, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1573. Available at: `https://aclanthology.org/D19-1573`.

Libovický, J. – Helcl, J. Attention Strategies for Multi-Source Sequence-to-Sequence Learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, p. 196–202, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2031. Available at: `https://aclanthology.org/P17-2031`.

Libovický, J. – Helcl, J. End-to-End Non-Autoregressive Neural Machine Translation with Connectionist Temporal Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 3016–3021, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1336. Available at: `https://aclanthology.org/D18-1336`.

Libovický, J. – Helcl, J. – Mareček, D. Input Combination Strategies for Multi-Source Transformer Decoder. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, p. 253–260, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6326. Available at: `https://aclanthology.org/W18-6326`.

Liwicki, M. – Graves, A. – Fernàndez, S. – Bunke, H. – Schmidhuber, J. A Novel Approach to On-Line Handwriting Recognition Based on Bidirectional Long Short-Term Memory Networks. In *Proceedings of the 9th International Conference on Document Analysis and Recognition, ICDAR 2007*, 2007.

Ma, X. – Zhou, C. – Li, X. – Neubig, G. – Hovy, E. FlowSeq: Non-Autoregressive Conditional Sequence Generation with Generative Flow. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, p. 4282–4292, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1437. Available at: `https://aclanthology.org/D19-1437`.

Miceli Barone, A. V. – Helcl, J. – Sennrich, R. – Haddow, B. – Birch, A. Deep architectures for Neural Machine Translation. In *Proceedings of the Second Conference on Machine Translation*, p. 99–107, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4710. Available at: `https://aclanthology.org/W17-4710`.

Popel, M. – Bojar, O. Training Tips for the Transformer Model. *The Prague Bulletin of Mathematical Linguistics*. April 2018, 110, p. 43–70. ISSN 0032-6585. doi: 10.2478/pralin-2018-0002. Available at: `https://ufal.mff.cuni.cz/pbml/110/art-popel-bojar.pdf`.

Popović, M. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, p. 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049. Available at: `https://aclanthology.org/W15-3049`.

Post, M. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, p. 186–191, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6319. Available at: `https://aclanthology.org/W18-6319`.

Qian, L. – Zhou, H. – Bao, Y. – Wang, M. – Qiu, L. – Zhang, W. – Yu, Y. – Li, L. Glancing Transformer for Non-Autoregressive Neural Machine Translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, p. 1993–2003, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.155. Available at: `https://aclanthology.org/2021.acl-long.155`.

Ran, Q. – Lin, Y. – Li, P. – Zhou, J. Guiding Non-Autoregressive Neural Machine Translation Decoding with Reordering Information. *Proceedings of the AAAI Conference on Artificial Intelligence*. May 2021, 35, 15, p. 13727–13735. Available at: `https://ojs.aaai.org/index.php/AAAI/article/view/17618`.

Rei, R. – Stewart, C. – Farinha, A. C. – Lavie, A. COMET: A Neural Framework for MT Evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 2685–2702, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.213. Available at: `https://aclanthology.org/2020.emnlp-main.213`.

Rozis, R. – Skadiņš, R. Tilde MODEL - Multilingual Open Data for EU Languages. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, p. 263–265, Gothenburg, Sweden, May 2017. Association for Computational Linguistics. Available at: `https://aclanthology.org/W17-0235`.

Schwenk, H. – Chaudhary, V. – Sun, S. – Gong, H. – Guzmán, F. Wikimatrix: Mining 135M Parallel Sentences in 1620 Language Pairs from Wikipedia. *arXiv preprint arXiv:1907.05791*. 2019.

Sennrich, R. – Haddow, B. – Birch, A. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 86–96, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1009. Available at: `https://aclanthology.org/P16-1009`.

Shao, C. – Zhang, J. – Feng, Y. – Meng, F. – Zhou, J. Minimizing the Bag-of-Ngrams Difference for Non-Autoregressive Neural Machine Translation. *Proceedings of the AAAI Conference on Artificial Intelligence*. Apr. 2020, 34, 01, p. 198–205. doi: 10.1609/aaai.v34i01.5351. Available at: `https://ojs.aaai.org/index.php/AAAI/article/view/5351`.

Sun, Z. – Li, Z. – Wang, H. – He, D. – Lin, Z. – Deng, Z. Fast Structured Decoding for Sequence Models. In Wallach, H. – Larochelle, H. – Beygelzimer, A. – Alché-Buc, F. – Fox, E. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems*, 32, p. 3016–3026. Curran Associates, Inc., 2019. Available at: `https://proceedings.neurips.cc/paper/2019/file/74563ba21a90da13dacf2a73e3ddefa7-Paper.pdf`.

Tiedemann, J. Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, p. 2214–2218, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). Available at: `http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf`.

Vaswani, A. – Shazeer, N. – Parmar, N. – Uszkoreit, J. – Jones, L. – Gomez, A. N. – Kaiser, Ł. – Polosukhin, I. Attention is All You Need. In *Advances in Neural Information Processing Systems 30*, p. 6000–6010, Long Beach, CA, USA, December 2017. Curran Associates, Inc.

Voita, E. – Talbot, D. – Moiseev, F. – Sennrich, R. – Titov, I. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. Available at: `https://aclanthology.org/P19-1580`.

Wang, Y. – Tian, F. – He, D. – Qin, T. – Zhai, C. – Liu, T.-Y. Non-Autoregressive Machine Translation with Auxiliary Regularization. *Proceedings of the AAAI Conference on Artificial Intelligence*. Jul. 2019, 33, 01, p. 5377–5384. doi: 10.1609/aaai.v33i01.33015377. Available at: `https://ojs.aaai.org/index.php/AAAI/article/view/4476`.

Wu, Y. et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*. 2016, abs/1609.08144. ISSN 2331-8422.

Zhou, C. – Gu, J. – Neubig, G. Understanding Knowledge Distillation in Non-autoregressive Machine Translation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. Available at: `https://openreview.net/forum?id=BygFVAEKDH`.

List of Publications

Helcl, J. – Libovický, J. Neural Monkey: An Open-source Tool for Sequence Learning. *The Prague Bulletin of Mathematical Linguistics*. Apr 2017, 107, 1, p. 5–17. ISSN 0032-6585. 54 citations.

Libovický, J. – Helcl, J. Attention Strategies for Multi-Source Sequence-to-Sequence Learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, p. 196–202, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2031. Available at: `https://aclanthology.org/P17-2031`. 140 citations.

Miceli Barone, A. V. – Helcl, J. – Sennrich, R. – Haddow, B. – Birch, A. Deep architectures for Neural Machine Translation. In *Proceedings of the Second Conference on Machine Translation*, p. 99–107, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4710. Available at: `https://aclanthology.org/W17-4710`. 80 citations.

Helcl, J. – Libovický, J. – Kocmi, T. – Musil, T. – Cífka, O. – Variš, D. – Bojar, O. Neural Monkey: The Current State and Beyond. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, p. 168–176, Boston, MA, March 2018. Association for Machine Translation in the Americas. Available at: `https://aclanthology.org/W18-1816` 7 citations.

Libovický, J. – Helcl, J. – Mareček, D. Input Combination Strategies for Multi-Source Transformer Decoder. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, p. 253–260, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6326. Available at: `https://aclanthology.org/W18-6326`. 43 citations.

Libovický, J. – Helcl, J. End-to-End Non-Autoregressive Neural Machine Translation with Connectionist Temporal Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 3016–3021, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1336. Available at: `https://aclanthology.org/D18-1336`. 67 citations.

Chen, P. – Helcl, J. – Germann, U. – Burchell, L. – Bogoychev, N. – Barone, A. V. M. – Waldendorf, J. – Birch, A. – Heafield, K. The University of Edinburgh's English-German and English-Hausa Submissions to the WMT21 News Translation Task. In *Proceedings of the Sixth Conference on Machine Translation*. Association for Computational Linguistics, 2021. 0 citations.

Only publications relevant to this thesis are included. The number of citations was computed using Google Scholar. Total number of citations of publications related to the topic of the thesis (without self-citations): 391