FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Dušan Variš

Learning Capabilities of the Transformer Neural Networks

Institute of Formal and Applied Linguistics

Supervisor: doc. RNDr. Ondřej Bojar, Ph.D.

Study Program: Computer Science
Specialization: Mathematical Linguistics

Prague 2022

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

Prague, December 4, 2022 Dušan Variš

Title: Learning Capabilities of the Transformer Neural Networks

Author: Dušan Variš

Department: Institute of Formal and Applied Linguistics

Supervisor: doc. RNDr. Ondřej Bojar, Ph.D.,
Institute of Formal and Applied Linguistics

Abstract:

Although the contemporary neural networks, inspired by biological neurons, were able to reach human-like performance on many tasks in recent years, their optimization (learning) process is still very far from the one observed in humans. This thesis investigates various aspects of learning in the current state-of-the-art Transformer neural networks, the dominant architecture in the current neural language processing. Firstly, we measure the level of generalization in Transformers using several probing experiments based on the idea of adversarial evaluation. Secondly, we explore their potential for incremental learning when combined with regularization using the elastic weight consolidation approach. Lastly, we propose a modular extension of the existing Transformer architecture enabling subnetwork selection conditioned on the intermediate hidden layer outputs and analyze the attributes of this network modularization. We investigate our hypotheses mainly within the scope of neural machine translation and multilingual translation showing the limitations of the original Transformer and the elastic weights consolidation regularization while presenting promising results of the novel modular Transformer architecture.

Keywords: neural machine translation, catastrophic forgetting, modular neural networks, incremental learning, generalization

Název práce: Schopnosti učení neuronových sítí Transformer

Autor: Dušan Variš

Katedra: Ústav formální a aplikované lingvistiky

Vedoucí práce: doc. RNDr. Ondřej Bojar, Ph.D.,
Ústav formální a aplikované lingvistiky

Abstrakt:

Přestože současné neuronové sítě, inspirované biologickými neurony, byly v posledních letech schopny dosáhnout lidské úrovně na mnoha úlohách, proces jejich optimalizace (učení) je stále velmi odlišný od procesů pozorovaných u lidí. Tato práce zkoumá různé aspekty učení současných neuronových sítí Transformer, převládající architektury pro zpracování přirozeného jazyka. V první části zkoumáme úroveň generalizace v Transformerech pomocí analytických experimentů založených na myšlence adversariální evaluace. V části druhé pak zkoumáme jejich potenciál pro kontinuální učení s použitím regularizace založené na elastické konsolidaci vah. V závěru práce navrhujeme modulární rozšíření stávající sítě Transformer umožňující výběr podsítí podmíněný zpracovaným vstupem spolu s demonstrací vlastností této síťové modularizace. Naše hypotézy testujeme především v kontextu neuronového strojového překladu a vícejazyčného překladu, přičemž naměřené výsledky odhalují limity původního Transformeru i metody regularizace pomocí elastické konsolidace vah. Navíc prezentujeme slibné výsledky navržené modulární architektury Transformeru.

Klíčová slova: neuronový strojový překlad, katastrofické zapomínání, modulární neuronové sítě, navazující učení, generalizace

Acknowledgements

I want to thank my supervisor for his guidance during my doctoral studies and the wisdom he provided whenever I felt stuck with a research problem. I would also like to thank Milan Straka for our valuable discussions regarding the deep learning theory and architectural design. Besides, I thank to the rest of my colleagues at UFAL for creating a positive and accepting workplace that made my studies pleasant and memorable.

I am also thankful to my parents, my sister Zuzana and my girlfriend Sara for all the emotional and material support they provided me throughout my studies. I would not be able to achieve this without them cheering me on in times of professional and personal struggle and without them believing in my abilities.

Lastly, I am grateful for the funding provided by the following projects, which directly and indirectly supported the research in this thesis: GA ČR EXPRO NEUREM3 (Neural Representations in Multi-modal and Multi-lingual Modeling; Grant number: 19-26934X (RIV: GX19-26934X)), H2020 Bergamot (Browser-based Multilingual Translation; Grant number: 825303), H2020 SSHOC (Social Sciences & Humanities Open Cloud; Grant number: 823782), LINDAT/CLARIN (Jazyková výzkumná infrastruktura v České republice; Grant number: LM2015071, EF16_013/0001781 pro CEP) and SVV 260 575 (Teoretické základy informatiky a výpočetní lingvistiky).

Contents

1 Introduction

Throughout recent years, deep learning (DL) has been a dominant machine learning paradigm, achieving state-of-the-art (SoTA) results in many fields, including image captioning (Xie et al., 2017; Szegedy et al., 2016; Krizhevsky et al., 2012), object detection (Lin et al., 2020; He et al., 2016a; Redmon et al., 2016), natural language generation (Brown et al., 2020a; Devlin et al., 2019; Peters et al., 2018) or machine translation (Popel et al., 2020; Vaswani et al., 2017; Johnson et al., 2017). The origins of deep learning research can be traced back to the late 1950s to the invention of the perceptron algorithm (Goodfellow et al., 2016; Rosenblatt, 1958), which has been inspired by the structure of a biological neuron in the human brain. A decade later, the arrangement of these artificial neurons into layers gave birth to artificial neural networks in the form of multi-layered perceptron (Ivakhnenko and Lapa, 1973). Based on the Universal Approximation theorem (Hornik et al., 1989; Cybenko, 1989), even a shallow neural network (NN) with only a single hidden layer and a non-linear activation function can, in theory, approximate a wide variety of functions; however, in practice, such network can be infinitely large. The number of network parameters can be reduced by increasing the depth of the network (Goodfellow et al., 2016).

Still, it was initially challenging to optimize complex neural networks. This was due to the computational requirements and the lack of an effective algorithm for computing network gradients necessary for gradient-based optimization. The introduction of the back-propagation algorithm (Rumelhart et al., 1986) in the 1980s was an important stepping stone towards reaching the current SoTA deep neural networks, although it was not until the early 2010s when the cheap and computation-efficient

GPUs allowed its effective implementation. The following advances in the efficiency of computer hardware and distributed computing and the underlying mathematical theory then lead to the current deep learning models that often even surpass humans on a variety of tasks, spanning from video games to machine translation.

Given its original motivation in the biological brain and the fascinating practical results, it seems logical to consider current deep learning models as an important ingredient for developing artificial general intelligence (AGI). On the other hand, the current DL models lack in certain areas of what we consider intelligence, namely in their ability to learn. For example, it can be argued that the SoTA language models such as BERT (Devlin et al., 2019) or GPT-3 (Brown et al., 2020a) reach human-like performance mainly due to the sheer size of the model (hundreds of billions of trainable parameters) and the huge amounts of training data (hundreds of billions of tokens). On the other hand, their human counterparts can gain similar or better language modeling capability by only seeing a fraction of the training data throughout their lifetime,[1] suggesting limitations of the current learning process in DL. Similarly, even though the SoTA neural architectures can learn to play video games on a super-human level, they require much larger experience (i.e. the number of game instances played) to achieve that performance (Tsividis et al., 2017). Additionally, these deep learning models are usually very narrow, being trained to solve a single task or a set of tasks that are similar in their nature. Such tasks also have to be learned simultaneously to avoid the well-known problem of catastrophic forgetting (CF, French, 1999) that emerges when the tasks are trained incrementally one after another.

These and other inefficiencies of the learning process, while not being the only weakness of the contemporary DL models, seem like a serious obstacle in the way of developing AGI systems. Ideally, AGI should have the ability to update its knowledge in time as new information becomes available. In this thesis, we focus on identifying the magnitude of these learning deficiencies in the context of the Transformer sequence-to-sequence NN architecture (Vaswani et al., 2017). We chose this architecture due to it being the basis for many of the current SoTA natural language processing (NLP) models. Our main focus is to study architecture in the context of sequence-to-sequence generation, namely machine translation (MT). Although some of the mentioned DL problems were already studied in the previous work to a certain degree, they were mostly explored in the context of simple toy networks on standard benchmark tasks, mainly from computer vision, e.g. MNIST (Deng, 2012) or CIFAR-10 (Krizhevsky et al., 2009).

[1] An estimated number of words the average human speaks throughout their lifetime is roughly hundreds of millions of words (Brandreth, 1980).

1.1 Research Questions

This thesis focuses on studying the learning capabilities of the sequence-to-sequence Transformer architecture in the context of both classical (single-task) and multi-task model optimization. We study the multi-task learning (MTL) from both perspectives of joint learning (the training data for all tasks are available at the same time) and incremental learning (the network is being optimized for one task at a time, learning in a fixed order to solve each task).

We are interested in the optimization aspects such as generalization, exploiting the prior knowledge about the previously-learned (and related) tasks, and avoiding CF. Additionally, we aim to better utilize the network capacity by decomposing the layers of the network into submodules that can be switched on and off depending on the processed input.

We experiment with the Transformer NN architecture (simply referred to as Transformer) as our base research architecture because one, it is currently the SoTA architecture for the majority of NLP problems, and two, it provides a good basis for our module decomposition experiments by already offering a level of modularity in its multi-head attention (MHA) mechanism. Furthermore, the previous findings demonstrated the emergent specialization ability of some of the attention modules (Voita et al., 2019) while other works suggest that non-vital attention modules can be pruned from the final network without hurting the overall performance hinting at an uneven utilization of the available network capacity (Michel et al., 2019).

We investigate these phenomena in the context of Transformer sequence-to-sequence architecture and our proposed modular variant, identifying the scope of the problems and exploring the methods designed to alleviate the negative effects of these phenomena. More precisely, we aim to answer the following three main research questions:

Research Question 1: *What is the extent of the generalization ability of the current Transformer models?*

Although Transformers often surpass their predecessors (recurrent networks, convolutional networks) on many NLP tasks, the reasons behind this success are still being studied. For example, the extensive research on the multi-head self-attention suggests that Transformer can effectively learn to model the structure underlying the processed sentences. Furthermore, the experiments in neural machine translation (NMT) show that Transformers can handle well the translation of rare words or named entities, although this success can be also attributed to the currently popular use of subword tokenization methods such as byte-pair encoding (BPE) or SentencePiece (Kudo and Richardson, 2018; Sennrich et al., 2016b). On the other hand,

there is currently a debate in the NLP research community on whether Transformers suffer from a poor ability to model long-distance dependencies (LDD), similar to the other types of sequence-processing networks (Devlin et al., 2019; Al-Rfou et al., 2019; Dai et al., 2019). We aim to investigate the reasons behind this behavior and try to dispute this belief by showing a potential high tendency of Transformers to overfit to the training data target-side sentence lengths. Additionally, we want to open a discussion about the construction of the NLP dataset splits that could help to better estimate the generalization power of future machine learning models. To provide the answer to this research question, we break it down into the following sub-questions:

RQ1.1 *Are long-range dependencies the only reason behind the performance drop when translating long sequences?*

The current poor performance of the Transformer-based models when generating very long sequences is attributed mostly to the inability to capture long-distance dependencies. Even though, in theory, the self-attention mechanism enables the model to attend to any token within a sequence directly, the quadratic computation cost of the attention can limit its effectiveness. Furthermore, we argue that the increase in the number of attended tokens leads to a higher saturation of attention, i.e. irrelevant tokens introduce additional noise to the attention layer output. We demonstrate in Chapter 4, that the problems with modeling LDD in Transformers can be also attributed to their tendency to overfit to the length distribution of the training sequences. We investigate our hypothesis by not only showing poor generalization ability on the longer validation sequences but also on sequences shorter than those encountered during training.

RQ1.2 *Are the current NLP approaches to dataset splitting sufficient to properly measure the generalization of sequence-to-sequence models?*

When estimating the generalization power of a model using validation data, we want to ensure that there is little to no overlap with the training data while being faithful to the true, real-world data distribution. In Chapter 4, we explore whether the similarity between the training and validation sentences in terms of vocabulary distribution can lead to a skewed estimation of the model's performance. We remove the most similar training examples using two similarity metrics and measure the change in the performance of a model optimized using the training data with the removed training instances. We hypothesize that a larger drop in translation quality after removing similar training instances

should imply that the model exploits these similarities and does not generalize well. We perform experiments with vocabulary-based similarity metrics and show that the similarity in the training/validation vocabulary distributions does not lead to a smaller generalization in Transformers.

RQ1.3 *Is the ability to copy unseen words conditioned on the subword length of the tokens seen during training?*

The treatment of out-of-vocabulary (OOV) words has been a thoroughly studied problem in early neural sequence-to-sequence research. This problem was usually perceived as a trade-off between the vocabulary size and the vocabulary specificity – either including whole words (usually in a lemmatized form) in the model vocabulary resulting in the model's size increase or replacing OOV words with a placeholder token and replacing the placeholder during the output post-processing based on a set of transcription rules. An alternative approach, the character-level NMT, replaces the need for a large vocabulary by modeling the whole sentence as a sequence of characters leading to an increase in the sequence length. Currently, subword tokenization became the most popular method for solving the OOV problem, enabling character-level representation of rare words while preserving longer and more complex vocabulary entries if they are frequent enough. The previous anecdotal evidence has shown that subword tokenization deals with OOV items through character-by-character transliteration, although to our knowledge there is no prior work analyzing the limits of this transliteration in NMT. We show in Chapter 4 that the accuracy of named-entity translation via copying during test time is closely related to the subword length of tokens seen during training. The presented results suggest that Transformers lack the ability to generalize to this translation rule.

Having shown some of the generalization problems of the contemporary Transformers, we shift our attention to another current machine learning problem, incremental learning. The main issue related to incremental learning is the lack of ability to retain previously learned knowledge, also known as CF. We focus on this phenomenon in the following research question.

Research Question 2: *Can selective parameter regularization lead to improvements in Transformer performance in incremental learning?*

Transformers, similarly to other neural network architectures trained using gradient-based optimization, suffer from CF when learning multiple tasks in a sequential order. One of the reasons is the difference between the loss spaces of two or more tasks where a low-error solution for Task A does not necessarily imply low error on the different Task B and vice versa. In joint MTL, CF is avoided by finding the optimum of the combination of the tasks, i.e. a low-error area in both loss spaces, usually by interleaving the training examples for both tasks. However, this requires the datasets for both tasks to be available at the same time which can be a problem for the long-term improvement of models in practice. A common approach is to opt for incremental learning where we fine-tune a current model with each new dataset. This leaves us with a separate model for each task (we keep a copy of the original model before fine-tuning). Recent work on CF proposed several approaches towards avoiding task interference, ranging from learning to generate replays from the previous tasks to applying parameter-specific regularization during the learning of the new task. We focus mainly on the selective parameter regularization approach, namely elastic weight consolidation (EWC), and analyze its effect on the incremental learning (IL) in Transformers. To answer the question, we divide it into the following sub-questions:

RQ2.1 *Can selective parameter regularization improve NMT performance when fine-tuning models initialized by pretrained language models?*

The problems with Transformer overfitting are even more apparent during NMT fine-tuning for the low-resource language translation. The lack of available good-quality parallel data is often solved by using secondary language resources such as monolingual corpora or parallel corpora in closely-related languages. The most common use of monolingual data is to create additional synthetic parallel data by back-translating the target-side language corpus. Compared to the data augmentation approach, the regularization methods include an additional training objective, i.e. language modeling objective, to force the model to retain knowledge about the previous language modeling task. Selective parameter regularization aims to achieve similar retention by identifying network parameters that are important for the previously learned task and penalizing the optimization algorithm for making large updates to these weights. We demonstrate in Chapter 5 that the selective parameter regularization of the Transformer decoder can lead to similar performance as the fine-tuning with language model (LM) regularization while requiring lower computation time.

RQ2.2 *How effective is selective parameter regularization for the different classes of incremental learning tasks?*

Based on the previously introduced IL classification task topology (Kemker et al., 2018), we propose its extension to sequence-to-sequence learning. In the context of the extended task topology, we propose simple string editing benchmarks which we use to evaluate the selective parameter regularization approach. In Chapter 5, we show which of the tasks can benefit from regularization We also discuss the limitations of the regularization method with respect to the previous work.

RQ2.3 *Does selective parameter regularization improve model performance in incremental multilingual translation?*

Similarly to the previous question, we measure the benefits in a multilingual NMT scenario. Our focus will be on multilingual transfer from the high-resource language pair to low-resource ones. In contrast to the simple string editing tasks, NMT models can exploit additional language similarities available in real-life scenarios. We compare the regularized model performance in Chapter 5 with both classical fine-tuning and joint training on all available tasks. We show that selective parameter regularization provides a trade-off between high-resource task knowledge retention and the ability to learn to translate low-resource languages at the cost of not excelling in either compared to the jointly learned counterparts.

After demonstrating the limitations of the selective parameter regularization in fine-tuning and its impact on the effective utilization of Transformer parameters in IL, we shift our focus to a different approach towards more efficient network capacity allocation. We analyze the possibilities of Transformer modularization in the final question.

Research Question 3: *Can Transformers benefit from the inclusion of conditional computation?*

Recent work in the area of neural network pruning has shown that after finishing the training process, a majority of the network parameters can be pruned (set to zero) without any major negative effect on the network performance, leading to very sparse final neural networks (Hoefler et al., 2021). Furthermore, this inefficient use of the available model capacity during training can make later scaling of the network architecture and storage of the resulting model problematic due to increased memory consumption. Even though the pruned network usually contains enough free parameters that can be utilized during the training on subsequent tasks, it can be challenging to effectively optimize these unused parameters without suffering from CF.

One option is to apply manual masking based on our notion of different tasks, however, this approach can introduce undesired inductive bias by making assumptions about the set of tasks. Thus, we aim at designing a modification to the contemporary Transformer architecture which includes mechanisms for conditional computation. We extend the Transformer with a controller sub-network that takes the input to its assigned Transformer layer and picks a module mask, actively selecting which parts of the layer should process the layer input. The controller behavior should be learned without supervision, allowing the Transformers to capture the underlying structure of the multi-task dataset while avoiding human-generated inductive biases. We are interested in the strengths and limitations of the modular Transformer, our interests are summarized by the following sub-questions:

RQ3.1 *What conditional computing approaches are suitable for Transformer modularization?*

We propose the modular modification of the Transformer with several variations. We compare these modular Transformer variations in Chapter 6 using mainly a simple string editing benchmark. Additionally, we evaluate the most promising modular Transformer configurations on the multilingual NMT to measure their applicability in more complex NLP tasks.

RQ3.2 *Can the Transformer modularization lead to the effective reduction of the active model parameters?*

Controller-based module choice, when properly regularized during training, can lead to the usage of a lower amount of network parameters when processing a given input. In Chapter 6, we measure the correlation between the reduction of the actively used parameters and the resulting model performance. We investigate the modular Transformer using both a simple string editing benchmark and the multilingual translation task. In addition, we measure the parameter reduction effect with respect to individual types of Transformer blocks, Transformer layers, and different training approaches.

RQ3.3 *Does Transformer modularization lead to task specialization of individual modules?*

In Chapter 6, we propose a metric based on conditional entropy to measure which modules get selected in connection to specific tasks in multi-task datasets and test whether a task specialization emerges in a modular Transformer. Besides task specialization, we also measure which modules become core for the Transformer network (being always selected) and which get activated only when certain conditions are met and how this behavior changes with respect to different types of Transformer blocks.

RQ3.4 *Which Transformer blocks are better suited for modularization?*

We investigate further in Chapter 6 the module selection behavior of different Transformer blocks. We use the previously defined metrics to see, which blocks are more likely to be pruned or contain modules that are always selected and which blocks contain more specialized modules. Besides comparing the behavior of different types of Transformer blocks, we also look at the modular Transformer block behavior with respect to the block depth.

1.2 Main Contributions

This section summarizes the main contributions of this to the study of the learning abilities of the Transformer models.

1.2.1 Algorithmic Contributions

1. We implement several approaches for creating adversarial evaluation datasets to study the generalization in Transformer NMT. This includes sequence-length generalization, dataset similarity exploitation, and rare-word translation (Chapter 4).

2. We implement the EWC regularization aimed at countering catastrophic forgetting in the neural sequence-to-sequence frameworks Neural Monkey and Fairseq. We also propose and implement an additional Fisher information normalization method and an alternative EWC regularization (Chapter 5).

3. We implement the modular Transformer architecture in Fairseq. Our implementation includes several modifications of the original architecture (Chapter 6). Notably, we implement the general module controller for conditional prediction of module masks using the Gumble-sigmoid prediction layer and a straight-through gradient estimation. The controller supports both token-level

and sequence-level mask prediction. Furthermore, we implement the masked multi-head attention that supports module selection, and similarly, we propose the modularization of the feed-forward network Transformer block. Lastly, we implement a training regularization that adjusts the ratio of the modules selected by the Transformer network while processing a given input (Chapter 6).

1.2.2 Empirical Contributions

We evaluate the Transformer generalization on the proposed adversarial datasets. In the multi-task and incremental learning experiments, we measure the performance of the proposed methods on both simple string editing datasets and the task of multilingual machine translation

1. We evaluate the ability of the original Transformer NMT to generalize to sequences that differ in length from the training data. Furthermore, we also measure the effects of the similarity between the training and test datasets on the skewness of the model performance estimation. Lastly, we show how the subword length distribution in the training data affects the model's ability to translate (copy) unseen words in the test data (Chapter 4).

2. We measure the effect of EWC on the unsupervised NMT model pretraining. Additionally, we analyze the limitations of the EWC regularization in the context of sequence prediction. We restrict the analysis to two variations of incremental task learning: a vocabulary expansion and true multi-task learning (Chapter 5).

3. We evaluate the performance of the proposed modular Transformer and how it is affected by the selection of the module budget training hyper-parameter. We compare the proposed architecture with the original Transformer on multilingual machine translation using both automatic metrics and a small-scale human evaluation. We also provide the analysis of the underlying module selection mechanism, measuring the module specialization in different Transformer layers.

1.3 Thesis Overview

Starting with the following chapter, the thesis is divided into six chapters. What follows is a short summarization of each of the chapters:

- Chapter 2 describes the multi-task learning problem and the closely related incremental learning problem. We provide a brief history of the research in these two areas and the current SoTA, with a higher focus on NLP-related research and NMT.
- In Chapter 3, we describe the key concepts tied to sequence modeling and sequence-to-sequence learning. Consequently, we provide a more detailed description of the Transformer network architecture, how it is optimized, and its inference-time applications. Lastly, we give a summary of the key concepts of NMT and the related evaluation methods
- Chapter 4 describes the experiments measuring several aspects of Transformer generalization using adversarial datasets. The experiments focus on the problems of length-related generalization, exploiting word distribution similarities between the datasets and rare word translation.
- Chapter 5 investigates the effects of EWC regularization in incremental learning, mainly with respect to catastrophic forgetting. We analyze the effects of EWC in unsupervised pretraining using monolingual data and in a more general multi-task incremental learning.
- We conclude the thesis in Chapter 7 summarizing our main findings.

1.4 Origins

This thesis contains the results from the following peer-reviewed publications, published by the thesis author. We use snippets from these publications throughout the thesis expanded by additional supplementary experiments conducted after publishing the original research publications.

- The main results on the length-based overfitting in Transformer in Chapter 4 are based on Variš and Bojar (2021), *"Sequence Length is a Domain: Length-based Overfitting in Transformer Models"*, published in EMNLP2021. Measured the ability of the Transformer to generalize to sequences of length varying from the training dataset sequence lengths. Discovered that Transformer is strongly biased towards generating sequences of lengths similar to those in the training data while disregarding the length of the test-time input.

- The experiments studying catastrophic forgetting in the unsupervised monolingual NMT pretraining are based on Variš and Bojar (2019), *"Unsupervised Pretraining for Neural Machine Translation Using Elastic Weight Consolidation"*, published in ACL-SRW2019. Compares the effects of EWC regularization on the process of fine-tuning using the LM trained on the available monolingual data with the previously suggested LM regularization.
- The results of the modular Transformer experiments are scheduled for publication at the time of writing the thesis. At the moment, the author is in the process of preparing the publication.

2

Multi-task Learning

The current dominant line of research in deep learning (and by extension, in general machine learning) focuses on optimizing statistical models to solve a single specific task or a set of tasks that are closely related to each other. This research approach makes sense because the isolation of individual tasks and task-specific datasets allows easier empirical analysis and understanding of the studied statistical models.

However, when aiming at building a more sophisticated, general-purpose artificial intelligence, it is necessary to design systems that can solve a larger variety of different tasks. Designing such systems is the main focus of multi-task learning (MTL, Caruana, 1997; Zhang and Yang, 2017) and its sub-field of incremental learning (IL, Solomonoff, 1989; Chaudhry et al., 2018). Systems that are trained on multiple tasks simultaneously are not only more compact (due to parameter sharing), but they also help with the data sparsity problem and can even boost the performance on the individual tasks by learning representations that capture more general and sometimes complementary knowledge from the given set of tasks (Caruana, 1997). In the following sections, we provide a brief introduction to MTL and its more challenging subdomain, IL. Due to the complexity of the topic, we restrict the work in this thesis mostly to the context of supervised learning, although the discussed topics can be also adapted to unsupervised or reinforcement learning.

We provide a summary of the research in MTL, namely the applications to tasks related to NMT. Our summary is based on the previous surveys of the works within the MTL community (Zhang and Yang, 2017; Chen et al., 2021b) which provide a wider summary of the recent trends in the general MTL research and the context

of NLP, respectively. Next, we shortly describe the problems related to IL and its current state of the research. Similarly, we base the provided description on a more extensive previous survey by Biesialska et al. (2020). Our focus will, however, be mainly on the context of sequence-to-sequence processing and NMT.

2.1 Multi-task Learning

The main problem with single-use statistical models is their narrow area of application. Furthermore, it is well-established that parameter initialization can have a strong effect on the training dynamics (Goodfellow et al., 2016). This need for proper parameter initialization became the basis for transfer learning (Thrun and Pratt, 1998; Pan and Yang, 2010; Zhuang et al., 2021).[1] The basic intuition behind it is that a model previously trained on an initial Task A has learned to produce useful feature representations that can become a good basis for the initialization of another model optimized for a subsequent Task B. However, this initialization-only knowledge transfer still expects storing a separate model for each new task, leading to storage requirements increasing with the increasing number of tasks, because without regularization, the optimization algorithm for the subsequent tasks will always try to find the best fit (i.e. overfit) for the task at hand.

MTL goes a step further: it aims at building models that extract knowledge about multiple tasks at once, each task is influenced by inductive biases provided by the remaining tasks, i.e. the other tasks are providing a form of regularization (Collobert and Weston, 2008; Guo et al., 2018; McCann et al., 2018). Additionally, previous work on neural network pruning shows that at the end of the network optimization, a large portion of the network's parameters can be removed without significantly affecting the network performance (LeCun et al., 1989; Hassibi and Stork, 1992), suggesting that there is an unused network capacity in the optimized network.[2] Although the number of unused parameters can be effectively reduced using knowledge distillation methods (Bucila et al., 2006; Hinton et al., 2015) most often characterized by the teacher-student learning framework (Tarvainen and Valpola, 2017; Kim et al., 2019), more effective utilization of the original network capacity could remove the necessity of these distillation approaches, reducing the training time and complexity.

[1] In the literature, transfer learning is sometimes considered synonymous with incremental or lifelong learning. We use this term to refer only to a model initialization using the previously optimized model without the need for knowledge retention.

[2] In this context, we use the term capacity loosely due to the lack of a clear consensus within the deep learning community about its definition. Recent works on network optimization suggest that a simple indication of the number of trainable parameters is insufficient in describing the effective network capacity in the traditional machine learning sense (Zhang et al., 2017, 2021b).

Besides, MTL comes up as a natural approach to solving complex tasks for which the training data requirements grow with the task complexity – more complex tasks usually require larger feature spaces for their proper representation leading to the well-established *curse of dimensionality* (Bellman et al., 1957), a phenomenon where increasing the size of representation requires a significantly larger increase in the training data for effective optimization. A logical solution to this problem is breaking a complex task into simpler sub-tasks, learning to find a solution to each sub-task in isolation, and recombining the individual solutions to solve the original task. However, solving the sub-tasks in isolation can easily disregard possible relations between them, and by learning to solve these sub-tasks jointly using MTL methods, these relations can be effectively exploited; a typical example being the object detection task which is usually modeled as separate object localization (marking the location of interesting objects within the image via bounding boxes) and image classification (identifying the object types within the image) task, both using a shared representational output of an image processing network (Ren et al., 2015; Redmon et al., 2016).

Thus, the main motivations behind MTL are: (1) exploiting the additional auxiliary task training data for individual task performance improvement while (2) producing a more compact and robust solution for a given set of tasks (compared to single-use models trained for each task separately).

In this thesis we use the term *homogenous* MTL when we refer to approaches that share all of the network parameters, implying that these tasks also share the same set of output labels and the main difference between the tasks is in the input-output distributions of their respective representative datasets. In the literature, this is sometimes referred to as multi-domain training (Bickel et al., 2007; Dredze and Crammer, 2008; Daumé III, 2009). An example of homogenous MTL can be the current state-of-the-art (SoTA) general-domain NMT models (Akhbardeh et al., 2021b) or certain multilingual NMT approaches (Yamagishi et al., 2016; Johnson et al., 2017).

We refer to MTL as *heterogeneous* if only a part of the network is actively shared between the tasks and some network layers are task-specific by design. In this case, the most common focus in NLP is on learning shared representations, often in the form of contextual embeddings (Peters et al., 2018; Devlin et al., 2019; Brown et al., 2020b) that combine the knowledge about every task involved while the task-specific layers are optimized to operate over these shared representations. Importantly, the choice of task-specific network modules is strictly tied to the task at hand. The give an example, we consider the multilingual NMT approach presented in Bapna and Firat (2019) heterogeneous because the language-specific adapters (additional feed-forwards layers) are chosen explicitly by the user based on the currently processed language pair. In contrast, we consider the work presented by Zhang et al. (2021a) as

homogenous MTL because the whole network is shared between all the languages, and the choice of a suitable adapter is done by the trainable gating mechanism within the network. Although possible, the latter approach does not guarantee the emergence of strictly task-specific network modules. Still, based on their empirical evidence, some modules are more likely selected when processing specific languages.

Both homogenous and heterogeneous MTL have their own applications, although we argue that homogenous MTL is more desired because it helps to remove some inductive bias, i.e. explicitly stating which part of the network computation needs to be executed for a specific task, ignoring possible overlap between the tasks or whether they complement each other. On the other hand, when working with heterogeneous inputs, e.g. training a joint model for image captioning and machine translation (MT), having a separate image encoder for the former and a text encoder for the latter, is required due to the different format of the inputs, each requiring a different pipeline of transformations into a shared representation space (Kim et al., 2020; Jia et al., 2021; Ni et al., 2021).

The biggest contribution to the increase in the popularity of MTL within the NLP community can be most likely accredited to the introduction and increasing accessibility of the large language models (Devlin et al., 2019; Brown et al., 2020b) and their various flavors (Conneau and Lample, 2019b; Liu et al., 2019; Martin et al., 2020; Xu et al., 2021). These models, often trained on hundreds of billions of training tokens using mainly a masked language model (MLM) objective (Devlin et al., 2019) as their core optimization objective, provide high-quality, generalized, contextual embeddings that can be later used for fine-tuning models for various downstream tasks, such as question answering (QA), natural language inference (NLI) or even NMT. Most of the time, the original LM with its parameters fixed becomes part of a network that is shared between the tasks, and task-specific output layers are added and optimized on the respective task-specific data. The choice between fixing the shared layers or further fine-tuning these layers during the down-stream task learning differs between applications; while fine-tuning usually leads to representations that are better suited for the end tasks, it can lead to over-fitting in cases when only limited data are available for the downstream task (Peters et al., 2019; Grießhaber et al., 2020). In the latter case, extraction of the LM features and using them as input for the downstream task classifiers can be a more feasible option that prevents overfitting.

A slightly different approach to MTL can be seen in the area of multilingual NMT. The standard encoder-decoder NMT paradigm (Bahdanau et al., 2014; Ha et al., 2016) allows both parameter sharing and language-specific sub-networks or a combination of these two approaches. In general, multilingual NMT can be classified into three categories: many-to-one, one-to-many or many-to-many.

Many-to-one NMT systems (Lee et al., 2017) focus mainly on learning general, language-independent input representations that can be subsequently translated to the target language by the decoder network. Such an approach can be beneficial when translating from a low-resource source-side language by combining the under-resourced datasets with data available for, ideally closely related, high-resource source-side languages (Tars et al., 2021b). The high-resource languages then work like a form of regularizer, not allowing the network to overfit to the low-resource data. The introduction of subword embeddings (Sennrich et al., 2016b; Kudo and Richardson, 2018) allows even the combination of more distant languages without the need to significantly expand the model's vocabulary, at least when the languages share the writing script (Kocmi and Bojar, 2018). While not explicitly a many-to-one NMT instance, the use of backtranslation (Sennrich et al., 2016a) can be considered a form of multi-domain NMT, if we consider the synthetic sentences created by automatic translation from the target to source language as examples from a language that is very similar to the original source-side language. In this case, the synthetic task of learning to translate backtranslated source sentences to the target language serves as an auxiliary regularization task.

There are two general approaches to the *one-to-many* NMT paradigm. The simpler, but less storage-efficient approach focuses on training an additional decoder module for each new target language (Dong et al., 2015; Luong et al., 2016) and using the respective decoder when translating into a chosen target language. The resulting NMT system storage requirements grow linearly with the number of supported target-side languages. The system also enables only limited exploitation of the target-side language similarities. Still, it allows some level of regularization through the shared encoder via input representations produced by the encoder. To allow greater exploitation of the similarities between the target-side languages, only selected parts of the decoder can be declared as language-specific, resulting in a more compact model (Wang et al., 2018b; Bapna and Firat, 2019). Ultimately, a single-decoder solution is applicable, however, such an approach requires an explicit indication of which target-side language needs to be produced by the decoder, often by inserting a special target-side language indication token (Ha et al., 2016; Mhaskar et al., 2021).

The most complex, *many-to-many* NMT systems, combine the methods from the previous two multilingual NMT paradigms. While there are several ways of combining these paradigms (Johnson et al., 2017; Lu et al., 2018; Aharoni et al., 2019), the most interesting (and challenging) aspect of the many-to-many NMT is the zero-shot translation (Firat et al., 2016; Zhang et al., 2020). The idea of zero-shot translation is inspired by the notion of the *Vaquois Triangle* (Vauquois, 1968), based on the hypothesis that the multilingual word (subword) representations produced by the encoder are close to an interlingua, although, there is currently no clear consensus in the deep

learning community on the level of abstraction these representations provide. Using this shared interlingua, the many-to-many NMT decoder for a given target-side language should be able to decode the encoded sentence even if the system did not encounter the given source- and target-side language combination during the training. However, to achieve a good performance on the "zero-shot" language pair, at least a limited amount of fine-tuning is required (Firat et al., 2016). Recently, the pretrained, multilingual language models, such as XLM (Conneau and Lample, 2019a) or mBART (Liu et al., 2020), have been used as an initial source of such multilingual representations (Chen et al., 2021a, 2022).

2.2 Incremental Learning

Unlike the standard MTL, incremental learning[3] (IL, Solomonoff, 1989; Chaudhry et al., 2018) refers to the learning of a set of distinct tasks in a sequential manner. In addition to the problems common in the joint MTL, IL introduces new, unique challenges, namely catastrophic forgetting (CF), sometimes also referred to as catastrophic interference (CI, (McCloskey and Cohen, 1989)), making joint MTL generally easier to optimize in comparison. However, joint MTL requires data for all tasks to be available during the joint optimization which might not always be an option in practice, for example, due to the time-restricted licensing on certain datasets or the availability of pretrained models without access to the original training dataset. Furthermore, when maintaining statistical models in the long term, the original training data distribution used for the model optimization can likely be less descriptive of the real world which is continuously changing. This *concept drift* (Schlimmer and Granger, 1986) can then make the original model less accurate and requires either an update to the model or retraining it from the scratch. Different tasks in MTL may be subject to these drifts to different extents, making the adaptation more fragile.

The major problem accompanying incremental updates to deep learning models or their continuous adaptation to new tasks is CF. When a standard deep learning (DL) model, i.e. a model not explicitly designed to counter CF, is trained sequentially, the network tries to find the best fit for the currently presented dataset. While there might be a solution that fits both the current task and the previous task, the lack of previous task data in the current batches often results in the network *forgetting* the previous task by only fitting the current task data. To counter CF, we aim to design models with good knowledge retention. CF is also closely connected to the idea of model plasticity (Aljundi et al., 2019; Sodhani et al., 2020), the ability of the model to

[3]In the literature also often referred to as lifelong learning (Silver and Mercer, 2002; Silver et al., 2013; Parisi et al., 2019; Aljundi et al., 2017), continual learning (Ring, 1995) or sequential learning (McCloskey and Cohen, 1989; Shin et al., 2017).

effectively adjust its parameters when adapting to new information about the world. Eventually, a fixed-size model will reach a point when no new knowledge can be stored – we then say that the model reached its representational capacity (Biesialska et al., 2020). Such a model then must be either expanded by adding new layers or additional parallel modules (adapters; Bapna and Firat, 2019; Zhang et al., 2021a) or some selective forgetting must be applied, leading to a trade-off between the amount of past and present knowledge that the model stores, in the literature referred to as a stability-plasticity dilemma (Abraham and Robins, 2005).

In addition to mitigating CF, another goal of IL research is, similarly to joint MTL, to improve the performance on a specific task by using knowledge learned from other (often related) tasks. In this case, due to the incremental nature of IL, we distinguish between forward and backward knowledge transfer. Forward transfer aims at reusing past knowledge for improving future task performance (or learning). Backward transfer, on the other hand, happens when the model can improve its performance on previously learned tasks using the knowledge gained when learning newer tasks.

The contemporary IL research in NLP focuses mainly on the adaptation and fine-tuning, leveraging models pretrained on large, general-domain corpora (Devlin et al., 2019; Brown et al., 2020a), or models optimized on high-resource languages to improve performance on the under-resourced languages (Grießhaber et al., 2020; Thillainathan et al., 2021; Grießhaber et al., 2020; Kocmi and Bojar, 2018). While the already powerful contextual representations provided by the large LM trained on the unsupervised data can already help achieve high performance on various supervised downstream tasks (Brown et al., 2020a), subsequent domain-adaptation and task-adaptation of these LMs can lead to further performance gains (Gururangan et al., 2020). The earlier LM-based attempts at tackling the problem of CF were explored mainly on the replay-based methods (de Masson d'Autume et al., 2019b; Sun et al., 2020), generating pseudo-samples of the previously learned tasks during the subsequent learning.

Similarly, the studies of IL in NMT initially focused on adaptation of the existing high-resource and general-domain models by fine-tuning these models using the available in-domain low-resource corpora (Luong and Manning, 2015; Freitag and Al-Onaizan, 2016; Chu et al., 2017). These methods mostly ignored the effects of CF, sacrificing some amount of the original domain knowledge for more domain-specialized models due to model overfitting (Khayrallah et al., 2018). As a result, the fine-tuning approaches require the storage of a separate model checkpoint for each task involved in the training pipeline, being storage-inefficient. More recent

approaches suggest reusing a significant portion of the original network and storing task-specific knowledge in swappable adapters, reducing the overall store requirements for a given set of tasks (Bapna and Firat, 2019; Zhang et al., 2021a; Pfeiffer et al., 2020).

The multilingual NMT can be considered a special case of domain adaptation with individual translation pairs and translation directions representing separate domains (Tan et al., 2019; Johnson et al., 2017; Ha et al., 2016). Even though these models suffer from CF when being adapted to new languages, the multilingual NMT models trained on a combination of high-resource languages can still be a powerful parameter initialization when training low-resource NMT systems (Thillainathan et al., 2021; Tars et al., 2021a). Besides using regularization techniques aimed at reducing CF, a certain level of forgetting can be also prevented by fixing the encoders/decoders optimized for the previous languages during new language training (Escolano et al., 2021).

A problem closely related to the continuous multilingual NMT, namely to the model extension to new languages, is vocabulary expansion. While the subword tokenization solves the out-of-vocabulary (OOV) problems when the given set of languages share a writing system, the subword vocabulary produced by the tokenization algorithm is still affected by the training corpus token distribution. In the classic MTL setting, the best subword tokenization results can be achieved by combining the corpora available for all the languages involved in the training (Kocmi and Bojar, 2018). However, this is not possible during IL and the model needs to either reuse suboptimal subword tokenization or the model's vocabulary needs to be transformed with each new language (Kocmi and Bojar, 2020).

In the next chapter, we will introduce the basic concepts of sequence and sequence-to-sequence learning. We will also provide a summary of the Transformer neural network architecture.

3

Neural Sequence-to-sequence Modeling

Neural sequence-to-sequence learning is a set of tasks closely related to neural sequence modeling (Mikolov et al., 2010; Sundermeyer et al., 2012; Cho et al., 2014b; Bahdanau et al., 2014), more precisely, it can be considered conditional sequence modeling.

Let Σ be a finite alphabet and Σ^* the set of all possible strings over the alphabet. A neural sequence model is a function $f_\theta(\boldsymbol{w})$ that estimates the probability distribution $p(\boldsymbol{w}) = p(w_1, .., w_n), \boldsymbol{w} \in \Sigma^*, n \in \mathbb{N}$ over all possible sequences.[1] In practice, we often only estimate probabilities of sequences from a given language $\boldsymbol{L} \subset \Sigma^*$ setting $p(\boldsymbol{w}) = 0$ for $\boldsymbol{w} \notin \boldsymbol{L}$.[2]

In practice, most often due to the sparsity of the available data, estimating $p(w_1, .., w_n)$ directly is infeasible. Instead, we apply the chain rule and try to estimate the conditional probability of individual symbols given their predecessors:

$$p(\boldsymbol{w}) = \prod_{i=1}^{n} p(w_i | w_{i-1}, .., w_1) \tag{3.1}$$

The factorization in Equation 3.1 leads to a class of so-called autoregressive (AR) models. Before the recent advancements in deep learning, the most popular AR models were based on the hidden Markov model (HMM, van den Bosch, 2017). However, the HMMs are by definition built with a strong independence assumption, conditioning the output only on the predecessor sequence of limited length. On the other

[1] We will use the term sequence and string interchangeably.

[2] Based on the implementation, it can be either hard-coded in the function f_θ or inferred from the data, resulting in values very close to 0.

hand, even the simplest sequence modeling architectures can in theory make use of a very long sequence of predecessors, although, in practice, it is very hard to optimize such models for long sequences without additional modifications to the NN architecture (Kolen and Kremer, 2001). Additionally, while AR models are a very popular approach to modeling language in NLP, contemporary research suggests other alternative factorizations that aim to capture different dependencies within the sequence structure. Such approaches include (not exhaustively): right-to-left modeling that is often used for rescoring the output of the left-to-right AR model (Grundkiewicz and Heafield, 2018), non-autoregressive (NAR) modeling which assumes complete independence between the output symbols (Libovický and Helcl, 2018; Gu and Kong, 2021) or tree-based modeling aimed at mimicking the generation process given by a grammar accepting the language $\boldsymbol{L}$ (Gū et al., 2018). If not stated otherwise, this thesis will be centered around left-to-right AR architectures.

Let us assume two languages, $\boldsymbol{L_1} \subset \boldsymbol{\Sigma}^*$, and $\boldsymbol{L_2} \subset \boldsymbol{\Sigma}^*$. The goal of neural sequence-to-sequence modeling is to estimate a conditional probability of the output sequence $\boldsymbol{y} \in \boldsymbol{L_2}$ of length m, given the input sequence $\boldsymbol{x} \in \boldsymbol{L_1}$ of length n.

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{m} p(y_i|y_{i-1}, .., y_1, x_m, .., x_1) \tag{3.2}$$

Similarly to sequence models, the Equation 3.2 describes one of the possible factorization, a factorization that will be used throughout this thesis. However, the contemporary neural sequence-to-sequence models make several adjustments to this factorization leading to better empirical results. We will describe examples of these architectural designs in the next section.

3.1 Architecture Overview

The goal of the neural sequence generation is to output a sequence of length m, $\boldsymbol{y} = (y_1, .., y_m)$. In practice, this is achieved via a generator function that implements the probability distribution from Equation 3.1 and a decoding algorithm that samples the sequence from the modeled distribution. While there are multiple generation schemes, this thesis focuses mainly on the auto-regressive generation that uses the following (generalized) function:

$$y_i, h_i = f_{\boldsymbol{\theta}}(y_{i-1}, h_{i-1}) \tag{3.3}$$

The desired behavior of the function $f_{\boldsymbol{\theta}}$ with trainable parameters $\boldsymbol{\theta}$ is to successfully keep track of the previous sequential context using an internal history h_{i-1} which is updated at each decoding step using the output from the previous step and adjust the output distribution used for generating y_i given said history. This can be accomplished in various ways depending on the choice of the underlying neural network architecture.

Historically the first approaches towards modeling the neural generation function $f_{\boldsymbol{\theta}}$ were based on recurrent neural networks (RNNs). Intuitively, these networks seem like a logical choice given that they were designed to process dynamic time-series recurrently. Similarly to Equation 3.3, they keep track of the sequence history using the internal recurrent state. The implementation can be as simple as in the Elman networks (Elman, 1990) where a new hidden state h_i is produced by an element-wise sum of linear transforms of the input vector y_{i-1} and the previous state vector h_{i-1} followed by a sigmoid non-linearity. The output vector y_i is then produced by an output linear transform followed by another non-linearity (sigmoid or softmax) that models the output distribution of the discrete variable y_i.

The main issue with Elman's *vanilla* RNN is related to the training methods used to optimize it. When optimized using a gradient-based backpropagation (Rumelhart and McClelland, 1987), a common approach to NN optimization, the Elman networks often suffer from exploding and vanishing gradient problems that are related to learning to model long sequences (Hochreiter et al., 2001). Simply put, when using a variation of the backpropagation algorithm, backpropagation through time (BPTT), the further the error signal has to propagate, the more it *vanishes* due to the repeated multiplication of the gradient of the non-linearities, leading to a decreasing magnitude of the gradient. The exploding gradient is a result of a product of derivations of different classes of functions with a similar effect. Only instead of decreasing the gradient magnitude with the increasing distance from the final token, the magnitude of the gradient increases greatly, resulting in training instability.

One of the proposed solutions to these two problems was the introduction of a gating mechanism, creating more complex recurrent units such as long short-term Memory (LSTM, Hochreiter and Schmidhuber, 1997) or gated recurrent unit (GRU, Cho et al., 2014a). With the introduction of the gating mechanism, these recurrent units could better control the information flow, deciding what information should be stored within the fixed-length vector h_i and what can be discarded. While they helped with reducing the impact of the vanishing and exploding gradient, they did not completely remove the problem.

Although, the networks described so far can learn sequence-to-sequence mapping tasks, in practice, they treat the input and output sequence as a single concatenated sequence of tokens, usually separated by a special separator token, processing the sequence similarly as a languages model would. The initial input sentence is first processed by the network accumulating the information about the input sequence in the hidden state h_i, sometimes referred to as working in the *encoding mode*. Later, after processing a special end-of-sequence (EOS) token, the network switches to the *decoding mode*, producing the output sequence in the AR fashion. This distinction between the *encoding* and *decoding* modes leads to a logical topological separation of the original network into the *encoder* and *decoder* sub-network. These sub-networks can, depending on architecture design, either consist of separate sets of trainable parameters or make use of parameter sharing.

Given the input sequence $\boldsymbol{x} = (x_1, .., x_m)$ the encoder-decoder network first encodes the input sequence into a sequence of hidden states $\boldsymbol{h_{enc}} = (h_1, .., h_m)$ using the encoder:

$$\boldsymbol{h}^{(\boldsymbol{enc})} = f_{enc}(\boldsymbol{x}) \tag{3.4}$$

Namely, in the context of RNNs:

$$h_i = \mathit{RNN}_{enc}(x_i, h_{i-1}) \tag{3.5}$$

The encoder output $\boldsymbol{h}$ is then processed by the decoder producing the output sequence $\boldsymbol{y} = (y_1, .., y_n)$:

$$\boldsymbol{y} = f_{dec}(\boldsymbol{h}^{(\boldsymbol{enc})}) \tag{3.6}$$

Similarly to the encoder, this is usually implemented in RNNs as a function that updates the decoder hidden state:

$$h_j = \mathit{RNN}_{dec}(y_{j-1}, h_{j-1}) \tag{3.7}$$

And a token emission function:

$$y_j = f_{out}(h_j) \tag{3.8}$$

To condition the decoder sequence generation on the input sequence $\boldsymbol{x}$, we initialize the decoder state h_0 using the hidden states $\boldsymbol{h}^{(\boldsymbol{enc})}$ produced by the encoder. Usually, the RNN decoder (Sutskever et al., 2014) does not use the whole sequence $\boldsymbol{h}$ because the length of the input sequences (and their encodings) varies between examples and the decoder requires a hidden state of a fixed size. Instead, the RNN decoder uses either the last encoder hidden state h_m, or performs a maximum or average pooling (Hubel and Wiesel, 1959; Scherer et al., 2010) over the sequence $\boldsymbol{h}^{(\boldsymbol{enc})}$ to

create the initial decoding hidden state h_0. Also, note that the encoding Equation 3.4 results in hidden states encoding only the left-side context. A more robust approach, a bi-directional encoding (Peters et al., 2018), also considers the right-side context by running additional encoding in the opposite direction, modifying Equation 3.5:

$$h_i = RNN_{enc\_backward}(x_i, h_{i+1}) \tag{3.9}$$

This operation is usually performed by a separate *backward* encoder. The initial decoder state is then created by a concatenation of the *forward* and *backward* decoder initial state, created from the forward and backward encoder states respectively.

The main problem with this basic encoding method is the information bottleneck created by condensing a variable-length sequence into a fixed-length vector (the initial hidden decoder state). Besides, the encoding scheme described in Equations 3.5 and 3.9 still suffers from vanishing/exploding gradient problems when faced with very long input sequences. While the latter can be partially alleviated by forwarding information via additional shortcut connections or skip connections (He et al., 2016b; Bishop, 1995; Ripley, 1996), the former is not so straightforward. Bahdanau et al. (2014) tackled the problem with fixed-length encoder output by introducing an *attention mechanism* over the encoder output states $\boldsymbol{h}$. This mechanism enables the decoder to equip its hidden state with information from the variable-length encoder output $\boldsymbol{h}^{(enc)}$, by introducing an attention context to the decoding step (Equation 3.7):

$$h_j = f_{dec}(h_{j-1}, y_{j-1}, c_{j-1}) \tag{3.10}$$

The *attention context* vector c_i is a weighted sum over the encoder states $\boldsymbol{h}$, usually computed with respect to the current decoding hidden state:

$$c_j = \sum_{k=1}^{m} e_k h_k \tag{3.11}$$

The attention weights e_i, sometimes called *energies* (Bahdanau et al., 2014; Luong et al., 2016), are a result of a softmax operation (McCullagh, 2019) over the transformed encoder hidden states. The transformation methods may vary, including (not exhaustively) a fixed linear transform (Bahdanau et al., 2014; Luong et al., 2016), to a scale-dot product (Vaswani et al., 2017). Using the attention context, each decoding state can be directly influenced by any of the encoded input tokens $\boldsymbol{h}^{(enc)}$. Additionally, the direct connection between the decoder states and the encoder outputs can help reduce the *information bottleneck* caused by the fixed-length decoder state (Tishby and Zaslavsky, 2015).

The introduction of attention context, besides improving the performance of the RNN-based encoder-decoder architectures, was followed by the investigation of other alternative sequence-to-sequence approaches. For example, Lee et al. (2017) suggested a hybrid encoder architecture that combines a convolutional neural network (CNN, Lecun and Bengio, 1995) to first transform the input sequence of characters using a small contextual window and then applied RNN to refine their contextual knowledge within the context of the whole sequence. Later, Gehring et al. (2017) proposed replacing the RNN completely with a deep convolutional network. Using stacked convolutions and the fact that contemporary graphical processors (GPU) are well optimized for the computation of the convolution operation, they achieved significant speed improvements while surpassing the translation quality of the previous RNN-based sequence-to-sequence architectures.

While Lee et al. (2017) focused on creating a hybrid encoder by a combination of two different types of networks other approaches studied modifications of specific network archetypes (e.g. RNN) with techniques introduced in their competitive alternatives. For example, Fung and Mak (2018) replace the standard attention mechanism in the LSTM sequence-to-sequence with multi-head attention (Vaswani et al., 2017). Bogoychev et al. (2020) suggest first training a feed-forward Transformer network (described in the following section) by an RNN-based decoder, trained using teacher-student learning, for a more effective test-time inference.

3.2 Transformers

Ultimately, Vaswani et al. (2017) expanded on the idea of refining the contextual information introduced in convolutional sequence-to-sequence networks, by replacing the CNN layers in the encoder/decoder strictly by a feed-forward network (FFN). Figure 3.1 illustrates the proposed architecture, Transformer. The architecture builds upon several key concepts that we describe in the following subsections.

Embedding layer

First, the input tokens $\boldsymbol{x} = (x_1, .., x_m)$, represented by one-hot vectors indicating their position in a given vocabulary $\boldsymbol{V}$, are transformed into their respective continuous representations $\boldsymbol{x'} = (x'_1, ..x'_m)$ (aka word embeddings; Mikolov et al., 2013; Harris, 1954) using the input embedding matrix $W_{emb} \in \mathbb{R}^{|V| \times d}$ (d being the size of the hidden dimension):

$$x'_i = x_i W_{emb} \tag{3.12}$$

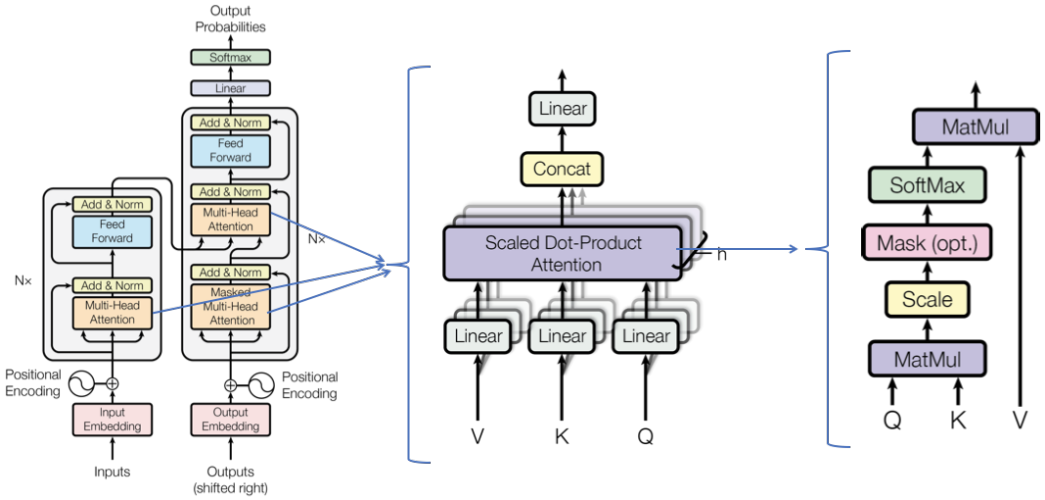

Figure 3.1: Illustration of the original Transformer NN architecture proposed by Vaswani et al. (2017). **Left**: General structure of the Transformer encoder-decoder architecture. **Middle**: General description of the Transformer multi-head attention block. **Right**: Detail of the scaled dot-product attention used in the Transformer attention blocks. The illustration was taken from the original publication.

The encoder and decoder network can either each have their designated embedding matrix, allowing the use of different source and target vocabularies, or both can share their embedding matrix parameters. While the former can help distinguish tokens with similar forms and different meanings between the source and target language, the latter results in a lower number of trainable parameters and can help to exploit lexical similarities between the source and target languages.

Similarly to the CNN-based architecture, the main Transformer blocks do not contain an explicit mechanism (such as the hidden state in RNN) for tracking the position of the tokens within a given sequence. Thus, the token embeddings $\boldsymbol{x}'$ are additionally shifted using the positional encoding (Vaswani et al., 2017) which is computed based on the absolute position of the tokens in the sequence. This positional shift of the embedding is computed as a dimension-specific combination of sine and cosine functions, mainly due to their periodic properties, proposed by Vaswani et al. (2017):

$$PE_{(pos,2i)} = sin(pos/10000^{\frac{2i}{d_{model}}}) \tag{3.13}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{\frac{2i}{d_{model}}}) \tag{3.14}$$

This leads to an updated embedding Equation 3.12:

$$x'_i = x_i W_{emb} + PE_{(i,*)} \tag{3.15}$$

Compared to the positional embeddings (Gehring et al., 2017), the benefit of the positional encoding is a fast computation (implementations of sine and cosine functions are well optimized in most of the deep learning frameworks and the results can be hashed due to the values being constant) and a reduction of the overall model size due to the positional encoding not requiring any additional trainable parameters.

The downside of the vanilla position encoding/embedding is its inability to explicitly model relative distances between the tokens. This results in the computational burden being shifted to the rest of the network.[3] There have been several proposals for introducing the representation of the relative token distance to Transformers, including a modification of the attention mechanism (Shaw et al., 2018) or a replacement of the positional encoding by a designated RNN layer (Neishi and Yoshinaga, 2019). These modifications showed promising improvements, mainly with respect to better modeling of long-range dependencies.

If not stated otherwise, we use the original Transformer architecture with the absolute positional encoding.

Transformer blocks

Following the embedding layer, the input tokens are processed by a stack of Transformer layers. Each Transformer layer contains multiple Transformer sub-layers, (refered to as blocks in this thesis; see Figure 3.1) – for the encoder, it is a self-attention block, followed by a feed-forward block, and for the decoder, it is the sequence of self-attention, encoder-decoder attention, and the feed-forward block. These blocks are the core of the architecture, performing step-by-step transformations of the input sequence representations while trying to capture the underlying structural elements of a sequence (source or target) that are important for learning to model the target sequence distribution.

A typical Transformer of depth K learns to apply functions $f^{(0)}, .., f^{(K-1)}$, represented by the respective Transformer blocks, each block producing a sequence of hidden representations $\boldsymbol{h}^{(j)} = (h_1^{(j)}, .., h_m^{(j)})$:

$$\boldsymbol{h}^{(j+1)} = f^{(j)}(\boldsymbol{h}^{(j)}) \tag{3.16}$$

[3] Although the solution is a simple algorithm (e.g. an absolute value of subtraction of two positional encodings), this ability is not default in a randomly initialized network and has to be learned from data. The original Transformer has a limited ability to correctly generalize when learning such algorithmic behavior as we demonstrate later in this thesis.

where $\boldsymbol{h^{(0)}} = \boldsymbol{x'}$.[4] Each block usually contains its own set of trainable parameters, learning a unique transformation of the input sequence. Some modifications suggest sharing parameters between the encoder and decoder (Rothe et al., 2020) to reduce the overall memory footprint of the model or even recursively applying a single Transformer block recurrently (Dehghani et al., 2019) which (based on the authors' claim) can enable higher expressiveness of the Transformer.

Figure 3.1 (middle and left) illustrates the multi-head attention mechanism. Vaswani et al. (2017) propose a general definition of the attention mechanism as a *"[...] mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vector"*. Previously, several types of attention mechanism were proposed (Bahdanau et al., 2014; Luong et al., 2016). The original Transformer uses a scaled dot-product product attention, also known as scaled bilinear attention (Luong et al., 2015b; Michel et al., 2019). Given a query q and a sequence of key-value pairs $(\boldsymbol{k}, \boldsymbol{v}) = ((k_1, v_1), .., (k_m, v_m))$, the single scaled dot-product attention is defined as:

$$\mathit{Att}(q, \boldsymbol{k}, \boldsymbol{v}) = W_o \sum_{i=1}^{M} \alpha_i (v_i W_v) \tag{3.17}$$

$$\alpha_i = \mathit{softmax}\left(\frac{q W_q W_k^\top k^\top}{\sqrt{d}}\right) \tag{3.18}$$

The trainable parameters in matrices $W_q, W_k, W_v \in \mathbb{R}^{d \times d}$ allow the model transform the query, key and value vector spaces in such a way that helps the attention mechanism to capture any structure within the sequence that is useful for a given task. Afterward, the result is further transformed using a trainable output matrix $W_o \in \mathbb{R}^{d \times d}$. The normalization constant $\sqrt{d}$ is used to preserve the unit variance of the representations after the dot-product operation (Vaswani et al., 2017).

Multi-head attention (MHA) expands this idea by learning N_{head} projections $W_q^{(j)}, W_k^{(j)}, W_v^{(j)}, W_o^{(j)} \in \mathbb{R}^{d \times (\frac{d}{N_{head}})}$ into (often smaller) vector sub-spaces (N_{head} indicating the number of attention heads), resulting in a respective $Att_j(.)$ attention head function. The MHA output is then defined as an ensemble of these individual attention outputs:

$$\mathit{MHA}(q, \boldsymbol{k}, \boldsymbol{v}) = \sum_{j=1}^{N_{head}} \mathit{Att}_j(q, \boldsymbol{k}, \boldsymbol{v}) \tag{3.19}$$

[4] Or, in the case of the decoder, $\boldsymbol{h^{(0)}} = \boldsymbol{y'}$

In practice, the transformations performed within the individual attention can be performed by combining respective query, key, value, and output attention matrices and executing a single matrix multiplication for each, leading to fast computation and parallelization. Furthermore, a sequence of queries $\boldsymbol{q} = (q_1, .., q_M)$ can also be processed in parallel.

In the context of Transformer, there are two applications of MHA: *self-attention* and *encoder-decoder attention*. The self-attention takes uses the input from the previous layer $\boldsymbol{h}^{(i)}$ in place of query, key, and value vectors, basically allowing the sequence $\boldsymbol{h}^{(i)}$ to *attend* over itself. The self-attention is the first processing sub-block within all of the encoder and decoder Transformer blocks. The encoder-decoder attention uses the previous decoder layer output $\boldsymbol{h}_{\boldsymbol{dec}}^{(\boldsymbol{i})}$ as the query and attends over the hidden states produced by the encoder final layer $\boldsymbol{h}_{\boldsymbol{enc}}^{(\boldsymbol{out})}$ which are used to derive both keys and values. The encoder-decoder attention is exclusive to the decoder, applied directly after the self-attention layer. The purpose of the encoder-decoder attention is to refine the decoder's hidden states using the knowledge about the source sequence extracted by the encoder.

Sometimes, we consider the MHA layer as means of *horizontal* transfer of information since the information stored in the hidden representations is transferred horizontally between the tokens in the sequence.

Feed-forward network (FFN) block applies a linear transform mapping the individual input vectors into a higher-dimensional vector space followed by a non-linearity, such as ReLU (Nair and Hinton, 2010), and another linear transform back to the vector space with the original dimension, using a pair of projection matrices $W_1, W_2 \in \mathbb{R}^{d \times 4d}$ and biases $b_1 \in \mathbb{R}^{4d}, b_2 \in \mathbb{R}^{d}$:[5]

$$\mathit{FFN}(h_i) = \mathit{ReLU}(h_i W_1 + b_1) W_2^\top + b_2 \tag{3.20}$$

Compared to the *horizontal* information transfer in the self-attention blocks, the theoretical purpose of the FFN blocks is to *vertically* propagate information between the dimensions of the individual hidden representations h_i.

Furthermore, each Transformer block applies a residual connection (He et al., 2016b) after each sub-block, connecting the sub-block output with its initial input, improving training convergence. Similarly, layer normalization (Ba et al., 2016) is applied between the individual sub-blocks for better training stability.

[5] The projection to a four times larger vector space is a general rule of thumb but different projection sizes can be chosen.

Output projection

Lastly, the output of the final Transformer decoder block is transformed by a linear projection mapping each hidden state h_i to the output token y_i. This projection is similar to the output projections used in other architectures (RNN, CNN), however, common practice is to replace the output projection matrix by the transpose of the decoder embedding matrix, tying the parameters of the decoder input and output projection (Press and Wolf, 2017).

This way, the output projection can be interpreted as generating the output token probability distribution (over the target vocabulary) based on the similarity of the hidden state h_i to the respective target-side vocabulary entries. During the inference we can either sample from this distribution or, more commonly, select a token that maximizes the probability of the generated sequence.

3.2.1 Transformer Training

The supervised training of Transformers (and other NMT architectures) aims at finding values of trainable model parameters (i.e. model weights) that minimize a training loss, most commonly the cross-entropy (CE) loss. Given the true conditional probability distribution $\hat{p}(\boldsymbol{y}|\boldsymbol{x})$ over the sequences $\boldsymbol{y}$ generated given the input sequence $\boldsymbol{x}$ and the estimated conditional probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ provided by the model decoder, the CE loss is defined as:

$$L_{CE}(\boldsymbol{\theta}) = -\sum_{\boldsymbol{x},\hat{\boldsymbol{y}}\sim\hat{p}} \hat{p}(\hat{\boldsymbol{y}}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\hat{\boldsymbol{y}}|\boldsymbol{x}) \tag{3.21}$$

Usually in practice, the true conditional probability distribution $\hat{p}$ is not available (or hard to track) because, in many NLP tasks, multiple output sequences can be often considered satisfying solutions (Bojar et al., 2013). Instead, we often have to rely on a single or only a few available true reference outputs. Thus, instead of minimizing the cross-entropy between the model distribution and the true data distribution, we aim to minimize the negative log-likelihood (NLL) of the true reference outputs with respect to the model distribution.[6,7] Let us assume that we train the model using data points $\boldsymbol{x}, \hat{\boldsymbol{y}} \in \boldsymbol{D}$, where $\boldsymbol{D}$ is our training dataset. By assigning sure event probability to each datapoint and using the cross-entropy (Equation 3.21), we arrive

[6] We aim for the maximum likelihood of our data, however in machine learning, the standard practice is to minimize the training loss.

[7] Throughout this thesis, we use the terms NLL and CE interchangeably due to their close relationship.

at the following NLL loss function:

$$L_{NLL}(\boldsymbol{\theta}) = -\sum_{\boldsymbol{x},\hat{\boldsymbol{y}}\in\boldsymbol{D}} \log p_{\boldsymbol{\theta}}(\hat{\boldsymbol{y}}|\boldsymbol{x}) \tag{3.22}$$

Following the factorization in Equation 3.2, we can modify the NLL loss, and instead of minimizing the NLL of the whole output sequence, we minimize the NLL of the individual tokens in the sequence given the current output prefix and the source sequence:

$$L_{NLL}(\boldsymbol{\theta}) = -\sum_{\boldsymbol{x},\hat{\boldsymbol{y}}\in\boldsymbol{D}} \sum_{i=1}^{|\hat{\boldsymbol{y}}|} \log p_{\boldsymbol{\theta}}(\hat{y}_i|\hat{\boldsymbol{y}}_{<\boldsymbol{i}}, \boldsymbol{x}) \tag{3.23}$$

Equation 3.23 implies an important thing: during training, the output of the i-th decoding step is conditioned on the previous $i-1$ golden truth tokens $(\hat{y}_1, .., \hat{y}_{i-1})$ instead of the actual decoder output $(y_1, .., y_{i-1})$. This technique, known as teacher forcing (Williams and Zipser, 1989), was first proposed for RNN training but found its application also with other sequence modeling architectures.

Although teacher forcing speeds up the model training convergence, it can lead to a phenomenon called *exposure bias* (Ranzato et al., 2016), i.e. during training, the model is only exposed to the gold output tokens (from the assumed true data distribution) when receiving input for the next decoding step. However, during test time, it is fully dependent on its learned token distributions. This can lead to a quick accumulation of errors when using simple decoding approaches such as greedy decoding, i.e. applying argmax operation on the output token distribution at each decoding step. The practical solution to this problem is using a more sophisticated decoding algorithm instead, the most common being *beam search* (Graves, 2012; Boulanger-Lewandowski et al., 2013; Sutskever et al., 2014; Freitag and Al-Onaizan, 2017). Another way to reduce the effects of the exposure bias can be *scheduled sampling* (Bengio et al., 2015), focusing on the teacher forcing at the beginning of the training and later, when the model is closer to convergence, switching to sampling from the learned distribution.

Another problem related to the NLL loss in Equation 3.23 is the fact that we maximize the likelihood of individual tokens instead of the whole sequences even though, at the end we evaluate the model against different criteria using different metrics that often operate either on the sentence-level or document-level, e.g. BLEU (Papineni et al., 2002) in NMT, ROUGE (Lin, 2004) in text summarization. To avoid the discrepancies between the training and test-time evaluation criteria, Shen et al. (2016) propose to use minimum risk training (MRT), a reinforcement learning (RL) technique useful for applying (often non-differentiable) metrics as the training loss. Another possible RL approach was proposed in the adversarial NMT (Wu et al., 2018),

replacing the reference-based training loss with an adversary neural network that is optimized to distinguish between the machine translation outputs and the human-produced reference translations, and letting the NMT generator and the adversary discriminator network that have opposing optimization objectives to *compete* against each other.

Compared to RNNs that contain an accumulator of the context information in the form of the recurrent hidden state, Transformers can further benefit from teacher forcing by efficient parallelization of the decoding. During training, decoding at all positions can be performed in parallel (instead of autoregressive decoding) because all decoder inputs are known in advance (gold reference tokens). There is no need for an accumulator due to the decoder's hidden states being the result of the stateless feed-forward and attention operations. Furthermore, the autoregressive-like property of the decoder (ie. only seeing the previous context during decoding) can be simulated by applying a triangular mask on the decoder inputs during training. This parallelization results in much faster training of the Transformer compared to the contemporary alternatives.

Besides the NN architecture modifications (residual connections, layer normalization) aimed at improving the stability of Transformer optimization, optimization algorithms that consider momentum, e.g. Adam (Kingma and Ba, 2014) or Adafactor (Shazeer and Stern, 2018) coupled with a specialized learning rate scheduling, including linear learning rate warmup and later exponential decay further improve Transformer convergence rate (Xiong et al., 2020; Vaswani et al., 2017). Additionally, label smoothing (Szegedy et al., 2016) applied during training can further help with the Transformer generalization ability.

3.2.2 Transformer Decoding

The optimized Transformer sequence-to-sequence model can be used either for sequence scoring, by computing the likelihood of the output sequence given the source input, or for conditional sequence generation. The latter, the autoregressive sequence generation, is a process of sampling output tokens given the source sequence and the previously generated output sequence prefix.[8] This process is repeated until a stopping criterion is met: usually either a special EOS symbol is generated or a maximum output length is reached. There are multiple algorithms for sequence sampling, the most common being greedy decoding, random sampling, and beam search.

[8] As mentioned earlier, there are other alternatives to autoregressive decoding, however, these are not covered by this work.

Greedy decoding (also known as argmax decoding) simply outputs the most probable token at each decoding step. Although quite fast and memory efficient, the resulting output hypothesis can be often suboptimal: even though the probability (likelihood) of each token is the highest at each step, the decoding algorithm can sometimes miss other solutions that can have a higher likelihood of the whole sequence (or subsequence), ie. the product of likelihoods of the greedily decoded tokens can be non-maximal.

Random sampling is often used in combination with other decoding algorithms mainly because of the conditionality of the output tokens on the available sequence prefix. The sampling itself is prone to generating erroneous outputs and the conditional nature of the decoder helps the errors to quickly accumulate. On the other hand, applying random sampling only during a few decoding steps can lead to hypotheses that can be overlooked by the deterministic decoding algorithms. Still, random sampling decoding is usually more commonly used during training, namely in combination with the optimization using reinforcement learning approaches (Wu et al., 2018; Shen et al., 2016).

Beam Search decoding was introduced as a more robust alternative to greedy decoding. It is designed to allow better exploration of the hypothesis space – instead of holding a single hypothesis prefix at a time (greedy decoding), beam search keeps track of K *working hypotheses* at each decoding step. When generating the next token, all potential partial hypotheses are expanded with every possible output token, scored, and only the new K best partial hypotheses are kept for the following decoding step. The scoring is usually based on the sum of the log-likelihood of the candidate's partial hypotheses tokens. By design, this scoring method often favors shorter sequences over longer ones (Stahlberg and Byrne, 2019). Therefore the scoring is modified by adding *length normalization* (Koehn and Knowles, 2017; Jean et al., 2015) or *length penalty* (Wu et al., 2016):

While, compared to the depth-first search algorithm, beam search is often not able to find the most likely hypothesis (according to the underlying model), it is still able to find hypotheses that are satisfying with respect to either automatic evaluation metrics and even human-based evaluation. This suggests that beam search can mitigate some biases contained in the underlying models (Stahlberg and Byrne, 2019).

The rest of this thesis will be covering our experiments with Transformer architecture, focusing on the problems of generalization, incremental learning, and Transformer modularization, answering the research question that we asked at the beginning of this work.

4

Generalization in NMT Transformers

One of the great research interests inside the machine learning (ML) community is the problem of building systems that learn and think similarly to humans, also referred to the artificial general intelligence (AGI, Lake et al., 2017). While the history of AGI research has been full of both optimism and disappointment, in the past decade, the interest in AGI research has grown large again, mainly thanks to important breakthroughs in the area of deep learning (DL), focused on studying large neural-based systems with multiple layers of representation.

This reawakened interest has also brought back to attention an old question: *What does it mean to learn and think like a human?* While the answer to this question is difficult and our current knowledge of the human mind is still not sufficient to provide a satisfying answer, it is still possible to compare certain aspects of the current state-of-the-art (SoTA) DL models to their counterparts in humans that serve often as a design inspiration for these models. Even though there are other non-neural approaches towards building an AGI, such as probabilistic ML (Ghahramani, 2015), automatic statistical reasoning (Lloyd et al., 2014), or probabilistic programming languages (Gelman et al., 2015; Goodman et al., 2008), our focus is set mainly on studying the aspects of learning in the context of DL, namely the Transformer architecture.

Lake et al. (2017) look at the task of building AGI as a combination of two key ingredients: implementing cognitive abilities into a system (such as intuitive physics and intuitive psychology as referred to by the author) and improving its ability to learn. While the former is no less important than the latter, our main area of interest is studying the aspects of learning with respect to Transformer architecture. Even

though Lake et al. (2017) state that the current DL systems are still far from implementing either of the key ingredients and compare the contemporary DL systems to *"models of statistical pattern recognition"*, we still find it relevant to talk about the ability to learn.

Recent advances in deep NLP (i.e. NLP with a focus on DL methods) resulted in models that can reach human-level performance in various tasks, spanning from *general* language modeling (Devlin et al., 2019; Brown et al., 2020a), to specific tasks such as speech recognition (Amodei et al., 2016; Chan et al., 2016), image captioning (Lu et al., 2017; Xu et al., 2015) or machine translation (Popel et al., 2020; Vaswani et al., 2017). Additionally, the previous trends of using rich linguistic knowledge to create detailed annotations of the training data describing the contemporary understanding of the language and its structure (Sudarikov et al., 2017; Bojar and Tamchyna, 2015) has been steadily on the decline thanks to recent DL advances. Instead of using human-based annotation DL models are able to identify and extract useful features with only little human guidance while also being able to scale with increasing amounts of training data (Brown et al., 2020a; Devlin et al., 2019).

While impressive on their own, these results fall a bit short when we focus on the comparison between these DL models and humans with regard to their learning process. A case can be made by comparing the human and SoTA DL ability to learn to play video games. For example, Mnih et al. (2015) introduced a deep reinforcement learning architecture based on Q-networks (Sewak, 2019) which was able to reach a human-level performance on the majority of selected Atari 2600 games without any specific knowledge about the individual games (the decisions were made based only on the visual input, i.e. what can be *seen* on the screen). While the performance of this architecture was a big step towards building independent agents it is important to acknowledge that the model required on average around 38 days of in-game experience (or 50 million training frames) and the models were highly specialized, training a separate model for each game. Compared to that, humans can learn to play these games in a matter of a few minutes (Tsividis et al., 2017) and are often not limited to a single game. This performance gap shows the huge difference between the complexity of contemporary neural networks and humans, possibly due to the latter having the advantage of having rich representations about objects and physics (Spelke, 1990; Baillargeon, 2004; Rips and Hespos, 2015), being able to acquire new concepts (Carey, 1978; Landau et al., 1988; Markman, 1991; Xu and Tenenbaum, 2007) and build intuitive theories (Murphy and Medin, 1985; Carey, 1985; Gopnik and Meltzoff, 1997), predict (Rips, 1975; Murphy and Ross, 1994) and imagine (Ward, 1994; Jern and Kemp, 2013).

Another case can be made using the recent Transformer-based LM, such as BERT (Devlin et al., 2019), its derivations, RoBERTa (Liu et al., 2019), Longformer (Beltagy et al., 2020) or Big Bird (Zaheer et al., 2020), or its upscaled version, GPT-3 (Brown et al., 2020a). These models are without doubt powerful language learners, often being used as an out-of-the-box *parameter initialization* that is subsequently fine-tuned for downstream tasks such as language understanding (Wang et al., 2018a), question answering (Rajpurkar et al., 2016; Joshi et al., 2017). However, the actual understanding of the language is still a subject of discussion. Recent criticism has compared these models to *stochastic parrots* (Bender et al., 2021), arguing that instead of language understanding, these models exploit the benefits of their scalability and the enormous size of the provided training corpora.

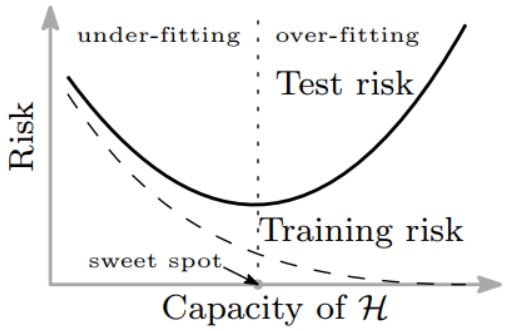

Figure 4.1: The classical U-shaped curve showing the trade-off between the model size and its ability to generalize on unseen data. We reproduce the original figure from Belkin et al. (2019).

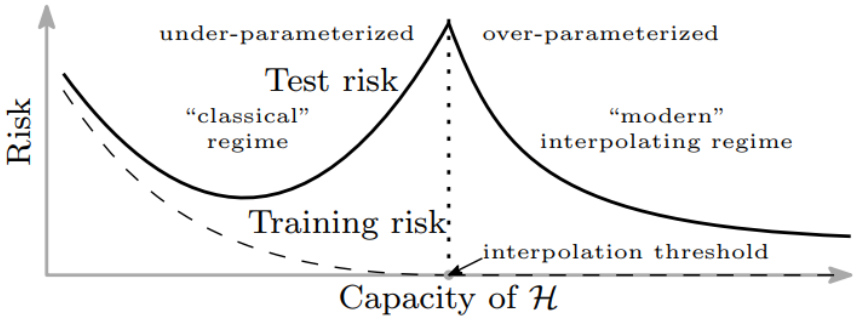

Figure 4.2: The double-descent curve proposed by Belkin et al. (2019). The curve still includes the previous U-shape curve, however, when the model becomes *over-parametrized*, its generalization error begins to decrease again (referred to as *interpolating regime*. We reproduce the original figure from Belkin et al. (2019).

If we look at the traditional machine learning theory, the action of model upscaling (e.g. by increasing the number of model parameters) is usually considered as a trade-off between the model expressiveness (i.e. ability to fit the training data) and generalization (having a low error rate on new data generated by the same dataset

distribution). This is often illustrated as a problem of finding an equilibrium between the model bias and variance (Hastie et al., 2001), illustrated by Figure 4.1). Based on this classical notion, an unbounded increase of the model complexity should eventually lead to a model with high generalization error (high difference between the train and test error rate), however, this does not seem to be the issue with the state-of-the-art deep NLP models. This goes against the previous machine learning intuitions based on the principle of Occam's razor (Gull, 1988).

On the contrary, recent work suggests that the classical U-shape (Figure 4.1) could be a result of not exploring model families that are *rich enough*. Belkin et al. (2019) show empirical evidence that as we further increase the model capacity, after a certain threshold, the generalization error starts to drop again (see Figure 4.2).[1] Even though all model classes past the *interpolation threshold* reach zero training error, the models from richer families achieve much better generalization error. They elaborate that the appropriate inductive bias for many of the real-world problems is the smoothness of the learned model function measured by a function space norm. Furthermore, Belkin et al. (2019) state that *"By considering larger function classes, which contain more candidate predictors compatible with the data, we are able to find interpolating functions that have smaller norm and are thus "simpler". Thus increasing function class capacity improves performance of classifiers"*. This in turn should work as an Occam's razor.

Surprisingly, the deep learning optimization algorithms are able to reach the solutions with low generalization error even without an explicit regularization (Zhang et al., 2021b) hinting at the *low norm* inductive bias being either an attribute of the optimization algorithm or the specific NN architectures. This tendency to prefer low-norm solutions could explain why is it possible to prune a majority of the model parameters after the model convergence (Michel et al., 2019; Li et al., 2020; Hoefler et al., 2021). Besides, recent evidence also suggests that stochastic gradient descent (SGD, Robbins and Monro, 1951) generalizes better in combination with over-parametrized models (Zhang et al., 2021b; Zhu et al., 2019; Jastrzebski et al., 2017; Hoffer et al., 2017).

Given the assumed interesting relation between the model over-parameterization and its ability to generalize, illustrated by the double-descent curve, it is important to be able to correctly estimate this generalization ability. The standard practice given a specific distribution generating data for our task is taking a small independent and identically distributed (IID) sample from the data distribution not present in our

[1] By the *capacity of a model* we mean the number of trainable weights, although, as Zhang et al. (2021b) point out, this might not be a correct complexity measure in the context of deep learning.

training dataset and use it to measure generalization error.[2] However, in the modern NLP governed by big data, the IID sampling is usually not the case due to training corpora being gathered from multiple sources, often independently from the annotated evaluation corpora.

For example, the current SoTA LMs consist of tens to hundreds of billions of trainable parameters (Goyal et al., 2021; Brown et al., 2020a) and are exposed to corpora gathered from various corners of the Internet,[3] observing hundreds of billions of words during training. These LMs are later evaluated on test suites, such as GLUE (Wang et al., 2018a) or SQuAD (Rajpurkar et al., 2016) that contain datasets created independently of the crawled training data, breaking (or, at least, muddying) the assumption of IID sampling from a true language distribution.[4] This gives rise to skepticism suggesting some level of undesired overlap between the train/test corpora leading to overestimation of model generalization ability. Using such LMs in combination with a limited understanding of the original unlabeled training data (which is not always publicly available, at least for commercial models developed and provided by big tech companies) can also result in wrong assumptions about the model capabilities and internal biases.[5]

For another example we can look at the News Translation Task from Conference on Machine Translation (WMT).[6] Each year, the source of training data for the featured language pairs are being updated, often containing corpora created automatically, or semi-automatically (e.g. by crawling bilingual websites and alignment of the collected text). The test corpora are, on the other hand, created by a collection of smaller samples of sentences from specific news sources and manually translated by professional translators (Akhbardeh et al., 2021a). Again, this breaks the IID sampling assumption of the classical machine learning evaluation of model generalization. While this was not a big issue a few years back before the emergence of big NMT models, we argue that in the era of overparametrized models, this could bias the evaluation and comparison of the recent NMT approaches.

[2] In practice, we usually work with multiple samples (e.g. validation, test, etc.) and use them during different stages of model development.

[3] A major fraction of the training data is made of the Common Crawl corpora created by the automated crawl of the websites from various domains. Common crawl is available at `https://commoncrawl.org/`.

[4] For example, GLUE contains questions from Quora and SQuAD contains passages from Wikipedia. In both cases, it can be assumed that the Common Crawl dataset also contains parts of these texts.

[5] For example, Abid et al. (2021) demonstrated that GPT-3 contains strong societal biases towards the Muslim minority.

[6] `https://statmt.org/wmt21/translation-task.html`

Although, there is a more general problem of independent creation of train/test datasets in some areas of modern NLP, there are areas that implement a more standard approach to evaluation. For example, areas such as image captioning or multimodal translation (e.g. MSCOCO; Lin et al., 2014) work with a single collection of annotated data that is later split into fixed train/valid/test datasets. These standardized splits (e.g. popular Karpathy splits for image captioning; Karpathy and Fei-Fei, 2015) better guarantee that the data for each split are generated from the same distribution and allow easier experiment reproduction and method comparison. However, a single fixed dataset split can still contain intrinsic biases that some methods can exploit better than others, easily leading to a Type I error (Gorman and Bedrick, 2019; Søgaard et al., 2020).

Such an issue can be partially solved by creating additional train/valid/test splits and tracking model performance under multiple different dataset splits (Gorman and Bedrick, 2019). This approach is currently applied in some NLP areas, such as sentence fusion (Geva et al., 2019) or visual pronoun coreference resolution (Yu et al., 2019). However, Søgaard et al. (2020) challenge this approach, arguing that random splits can still lead to overly optimistic performance estimates. Moreover, such estimates favor systems that overfit more – a behavior that is undesirable when comparing overparameterized models. Instead, they suggest focusing on creating more diverse testsets with different biases or splitting the available data in an adversarial way. We explore the idea of adversarial splits in this chapter to uncover some of the misconceptions about the generalization ability of Transformers.

This chapter will explore the benefits of using adversarial splits to identify shortcomings of the current Transformer models. First, we demonstrate the lack of generalization ability of the Transformers when modeling sequences of target-side lengths not present (or under-represented) in the training data. Next, we investigate, whether a textual similarity between the training and test data within a single domain can affect model performance, leading to an overestimation of the model performance on the said domain. Lastly, we present an experiment analyzing the effect of the byte-pair encoding (BPE) subword tokenization on the ability of the NMT model to copy named-entities that were previously unseen during the model training.

4.1 Sequence Length Overfitting

Previous experiments with Transformers in NLP (Popel et al., 2020; Brown et al., 2020a; Devlin et al., 2019) suggest that they accommodate strong generalization abilities. However, a closer-look analysis of these models suggests strong biases, sometimes leading to the use of foul and toxic language (Gehman et al., 2020) and per-

petuating negative stereotypes (Abid et al., 2021). Furthermore, Brown et al. (2020a) claim that their Transformer-based GPT-3 language model is capable, among others, of simple arithmetics, however, there is still an open discussion about whether this is due to the model actually (partially) understanding the underlying algorithms or it is just a result of sophisticated encoding a lookup table for the subset of arithmetic operation examples encountered during training.

In light of this argument, we present a set of experiments that demonstrate that the currently assumed generalization power of Transformers might be overestimated, being a result of the exploitation of the large volumes of training data and the model's ability to exploit the similarities between the available training and validation datasets. We show that they have a strong tendency to overfit to the training data, namely the decoder subnetwork in the Transformer sequence-to-sequence architecture.

This section presents results suggesting strong length-based overfitting in the Transformer decoder. While in theory, the Transformers should be capable of modeling long-distance dependencies (LDD), because the self-attention mechanism allows them to directly access any of the surrounding context of a given token in constant time, they can still struggle when faced with such examples in practice (Choshen and Abend, 2019). Still, we show that this might not necessarily be an inherent problem of Transformers, rather than a result of the insufficient number of training examples. The results of the following experiments show, that Transformers can not only struggle with modeling long sequences, but they also fail similarly on short ones if they are not presented with training examples of a similar length.

The main experiment results described in this section were previously published by Variš and Bojar (2021). Besides the summary of these results, we also present more recent follow-up experimental results.

4.1.1 Experiments

We demonstrate the limits of the length-related generalization ability of Transformers in two sets of experiments. The initial experiments investigate this behavior on simple algorithmic tasks of string editing and the follow-up ones demonstrate a similar behavior on a real-life machine translation (MT) task.

The ability of neural models to model very long sequences has been analyzed mainly with regards to a recurrent neural network (RNN). In this context, it was shown that besides the problems with exploding/vanishing gradients (Hochreiter, 1998; Bengio et al., 1994; Pascanu et al., 2013) the RNNs also suffer from a form an *information bottleneck* when trying to encode a dynamic-length sequence into a fixed-size vector (Bahdanau et al., 2014). To counter these weaknesses, Bahdanau et al.

(2014) introduced (with Luong et al., 2015a later expanding the idea) the attention mechanism to shorten the *information pathway* from the model output to the initial input, improving the effectiveness of the back-propagation algorithm used to update the network weights during training (Rumelhart et al., 1986).

Even though RNNs suffered from these restrictions, the recurrent hidden state displayed interesting properties, such as keeping track of the number of tokens already processed by the encoder or the number of tokens left to generate by the decoder (Shi et al., 2016). Unlike RNNs, Transformer, in its original formulation by Vaswani et al. (2017), does not contain such an internal tracking mechanism and is able to handle long sequences better than RNN nevertheless. The upside of not having an internal time-variant hidden state is better parallelism of Transformer architecture, mainly the Transformer encoder. On the other hand, due to its autoregressive nature, Katharopoulos et al. (2020) suggest replacing the dot-product followed by softmax (Equation 3.18) in the vanilla Transformer attention with the radial basis function (RBF) kernel attention (Tsai et al., 2019). This modification allows treating the Transformer decoder as a recurrent decoder with the hidden state being updated at each decoding timestep.

Lastly, a further look at the Transformer encoder and its self-attention mechanism that uses a non-sparse softmax suggests that it is lacking a mechanism that enables complete *forgetting* of irrelevant information, contrary to the gating mechanisms in the long short-term Memory (LSTM) or gated recurrent unit (GRU) recurrent networks. To tackle this, Correia et al. (2019) suggest a modified, sparse softmax operation as a replacement.

Mainly due to these dissimilarities between Transformers and its RNN predecessor, we focused on finding limits of the ability of Transformer to model dynamic-length sequences.

String Editing

In the initial experiments, we focus on training the Transformer to perform simple string editing operations. The main motivation behind this simple initial task is the much easier evaluation of the model performance. Compared to standard NLP tasks, string editing does not struggle with evaluation ambiguity (only a single correct output compared to multiple possible translations in MT). Furthermore, it allows for a straightforward dataset sampling resulting in balanced, non-overlapping training and validation data. This makes potential overfitting on the training data clearer, making it harder to exploit similarities between the training and validation data.

We defined the following set of editing tasks:

- **copy**: copy the input sequence to the output,

- **unshift X, push X**: add a single character (X) to the beginning or the end of the sequence, respectively,
- **shift, pop**: remove a single character from the beginning or the end respectively,
- **reverse**: reverse the character order in the input sequence.

Given the alphabet $\Sigma = \{a, b\}$, we generate the dataset by sampling a set of mutually distinct strings $\boldsymbol{x} \in \{x^k | x \in \Sigma, k < n\}$, where $n = 20$ is the upper bound on the string length. Next, we split the resulting data into bins based on the string length: 0–10, 11–15, and 16–20. From each bin, we independently sample 1,000 examples for test-time evaluation. Additionally, we sample 28,000 and 1,000 mutually distinct examples from the 11–15 bin for training and training-time evaluation, respectively.

Given these training and evaluation examples, we create the datasets for each string editing task by adding a token describing the operation to be applied, an *operation argument* token (a or b for unshift and push, $-$ for the others), and a separator token (|).[7] The gold target sequences are created algorithmically based on the input operation token. Examples of these data points are described in Figure 4.3.

We train a separate network for each task using the 11-15 bin dataset. For the string editing task, we chose Fairseq's `transformer` architecture (Ott et al., 2019) with the hyper-parameters being described in Appendix A.1.

Input	Output
push a | a b a b	a b a b a
reverse $-$ | a b b a a	a a b b a

Figure 4.3: Input and output example for `push` and `reverse` tasks. Hyphen ($-$) indicates an empty argument for the latter task. We reproduce the original figure from Variš and Bojar (2021).

During the evaluation, we measure model accuracy, $ACC = \frac{n_{correct}}{n_{all}}$, $n_{correct}$ indicating the number of exact matches between the network output and the reference string. Table 4.1 shows the accuracy of the models trained on individual tasks, showing their performance when applied to test datasets with a length similar to or varying from the original input data. The results show that the models generalize well when applied to new inputs from the same length distribution as the one in the

[7] The arguments for unshift and push are sampled from a Bernoulli distribution with a token having $p = 0.5$.

	0–10	11–15	16–20
copy	62.6	100.0	0.0
push	59.1	100.0	0.0
pop	0.1	100.0	0.0
shift	52.5	100.0	0.0
unshift	41.2	100.0	0.0
reverse	0.0	84.4	0.0
all	15.822	97.5	0.978

Table 4.1: Accuracy (in %) of models trained on various string editing tasks using only training data from the 11–15 length bin evaluated on datasets with different sequence lengths. Each model was evaluated on its respective task domain. We reproduce the original table from Variš and Bojar (2021).

training data, reaching perfect accuracy.[8] However, when faced with input strings with length outside of the training length distribution, the model performance drops significantly regardless of whether the input is longer or shorter than the training data.

A similar trend can be also seen when we train the model on the combination of the editing operations (`all` in Table 4.1). Although the model even learns to distinguish the operations based on the input operation label (with a small amount of error possibly due to the `reverse` operation), it fails similarly when faced with inputs outside of the training-length domain.

The results show that while it is not difficult to fit Transformer to the string editing tasks, it is unable to learn the general algorithm for completing these tasks.

Machine Translation

The next set of experiments investigates whether length-based overfitting occurs also during the training for the NLP tasks, specifically, machine translation. Compared to string editing, MT requires discovering more complex string mapping patterns; mappings between the input and output phrases are often heavily dependent on the surrounding context. Additionally, the distribution of input-to-output length ratios can be more varied. When combined with the need to learn complex relations between the tokens and their surroundings, could help the model to better generalize with regard to the length of the output sequence.

[8] The less than 100% accuracy for the *reverse* task is a result of the model not converging within the fixed limit of 100 training epochs and the difficulty of the sequence reversal task.

We test our hypothesis on English-Czech translation, using CzEng 2.0[9] (Kocmi et al., 2020) for training and WMT2020 (Barrault et al., 2020) `newstest13-20` for evaluation.[10]

We tokenize the corpora using Moses tokenizer[11] and apply subword tokenization using BPE (Sennrich et al., 2016b) based on the training corpus with subword segmentation of size 30k. Next, we split the datasets into bins of sentence pairs with lengths 1–10, 11–20, .., 91–100 (labeled as 10, 20, ..., 100 respectively), either based on the length of either the source-side or target-side sequence. Table 4.2 shows the sizes of the respective training corpora depending on the split criterion (source vs target).

Bin (Source-side)	0–10	11–20	21–30	31–40	41–50	51–60	61–70	71–80
# of sent. pairs (M)	30.9	15.3	6.5	3.5	1.9	1.1	0.6	0.4
# of tokens (M)	375.7	451.0	327.3	244.9	174.4	115.0	78.0	53.2
Bin (Target-side)	0–10	11–20	21–30	31–40	41–50	51–60	61–70	71–80
# of sent. pairs (M)	30.9	18.0	7.5	3.9	2.3	1.2	0.7	0.4
# of tokens (M)	375.3	502.6	361.6	268.9	198.9	132.6	87.3	59.5

Table 4.2: Sizes of the respective training bins (created based on either source-side or target-side sequence length) in millions of sentence pairs and millions of tokens (after tokenization and applying BPE, combined source and target size). We reproduce the original table from Variš and Bojar (2021).

Using each training bin, we train a separate model. Details about the model hyper-parameters are described in Appendix A.2.

First, we measure the correlation between the difference in train-test length of sentences and the resulting system performance measured by BLEU, specifically, the SacreBLEU implementation (Post, 2018).[12] Figure 4.4 (Top) shows that regardless of the training bin, the model performs best when presented with data that have target-side lengths similar to the length of the training data. This suggests that the model overfits to the length of the training data, affecting its performance when faced with either longer or shorter sentences. Although the BLEU scores are not directly comparable between the individual test bins (i.e. between individual *columns* in Figure 4.4, top), there is a suspicious correlation between the length difference in the train-test data and the performance drop. Figure 4.4 (Bottom) sheds some

[9] `https://ufal.mff.cuni.cz/czeng`

[10] We use `newstest13-16` for training-time evaluation (validation) and `newstest17-20` for test-time evaluation (testing). We use SacreBLEU (Post, 2018) to download the respective evaluation corpora.

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

[12] `https://github.com/mjpost/sacrebleu`

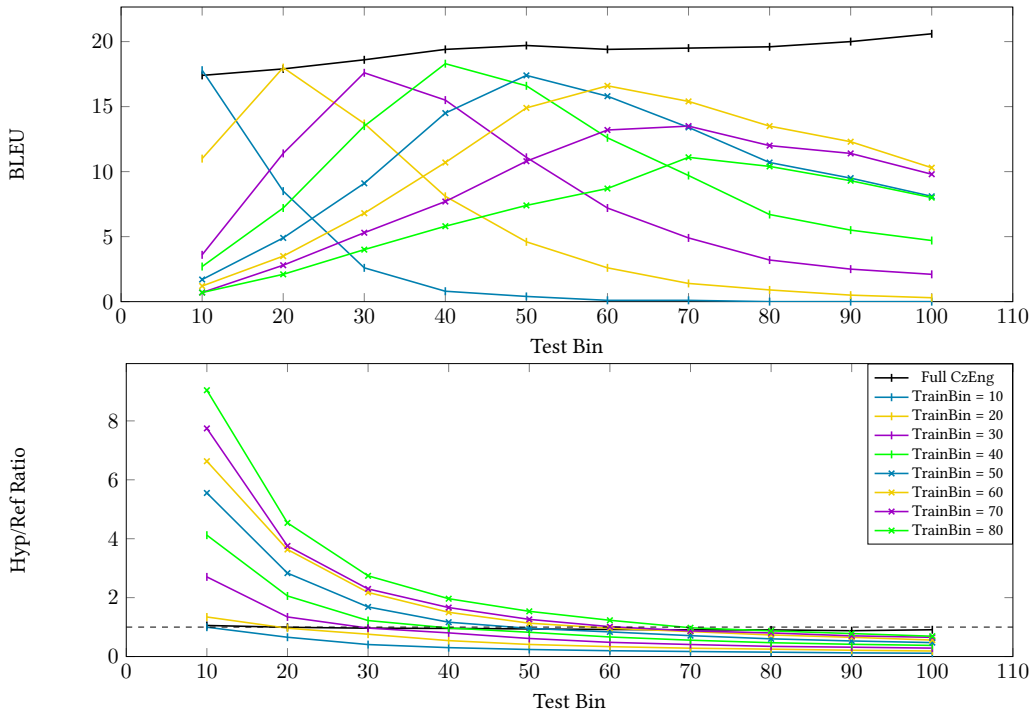

Figure 4.4: **Top**: Varying performance of Transformers on test data trained only on the data from a specific target-side length bin (various lines) when evaluated on a specific test bin (x-axis). When the train-test sentence length difference increases, the performance drops. Note that BLEU scores are not directly comparable across different test sets (i.e. horizontally). Within each test set, we see that the Full CzEng and the training bin of the matching length are the two best results. **Bottom**: Average ratio between a hypothesis and reference. Dashed line indicates a ratio of 1.0. Systems trained on short sentences produce short outputs, and systems trained on long sentences produce up to 10x longer outputs (Train bin 80 evaluated on Test bin 10). We reproduce the original figures from Variš and Bojar (2021).

light on the reason behind this drop in translation quality. When we compare the lengths of the generated hypotheses with the length of the reference translation we can see a strong bias towards generating hypotheses of lengths similar to the training data, further suggesting length-based overfitting on the Transformer decoder. Note that this overfitting trend is not as clear when evaluating using the 70+ length bins, possibly due to a low amount of training examples within the respective training data ($< 1M$ sentence pairs).

To confirm our hypothesis about decoder-side overfitting, we repeated the experiments using dataset splits based on the source-side length. Related work which focuses mostly on improving very long sentence translation (Shaw et al., 2018; Neishi and Yoshinaga, 2019) performs splits according to this criterion, motivated by the

real-life application – outside of closed-environment experiments, we do not usually have the reference translations available. Still, Neishi and Yoshinaga (2019) point out similar overfitting behavior of vanilla Transformers with regards to the translation of either longer or shorter test sentences.

Truly, Figure 4.5 shows that the overfitting trend can be observed even when we split our dataset according to the length of the source sentences. However, the trend is less clear than during the target-side dataset split experiments. To get more insight into the possible reasons behind the lesser clarity of the overfitting behavior, we examined the target-side length distribution within the individual source-length-based bins.

Figure 4.6 shows the target-side length distribution inside the individual source-length-based bins in the training (left) and test (dataset). Based on the distribution outliers, inside all training bins, there is a non-empty overlap between the target-side lengths of the training data and test data. This overlap is most likely behind the lesser clarity of the overfitting phenomenon when studied using source-side binning. The target-side length distribution analysis also suggests that CzEng 2.0 contains several *dirty* sentence pairs, where the target-side length is several times larger than the source sentence. Whether the removal of such incorrect translation examples can benefit the overall model performance or can lead to a decrease in the generalization ability of the model is left to future work.

As demonstrated by the results in Figure 4.4 (bottom) there is a clear tendency of the MT model to produce output hypotheses of lengths similar to the training data. Due to the nature of the used automatic metric (BLEU), this is most likely the reason behind the poor model performance when applied to much longer/shorter input sentences. Thus, we decided to perform a small sample case study of the translation outputs with regards to the train-test length difference.

Table 4.7 two selected examples of translation of sentences from the 30-bin using systems trained on shorter examples (10-bin), similar length examples (30-bin), and longer examples (60-bin). Although still producing some errors, the *in-domain* 30-bin system produces a translation hypothesis that is closest to the correct translation. On the other hand, even the other two systems (10-bin and 60-bin) produce semantically close translations of the input sentences. The main source of error is stemming from the systems trying to condense (10-bin) or stretch (60-bin) their translation hypothesis to fit the length range from their respective training data. This results in the 60-bin system generating repeating phrases (*v Žižkově, mluvčí Olomouckého*) or completely irrelevant outputs (*vojska*). Surprisingly, in the case of the *short* system (10-bin), its behavior slightly imitates a combination of translation and sentence summarization,

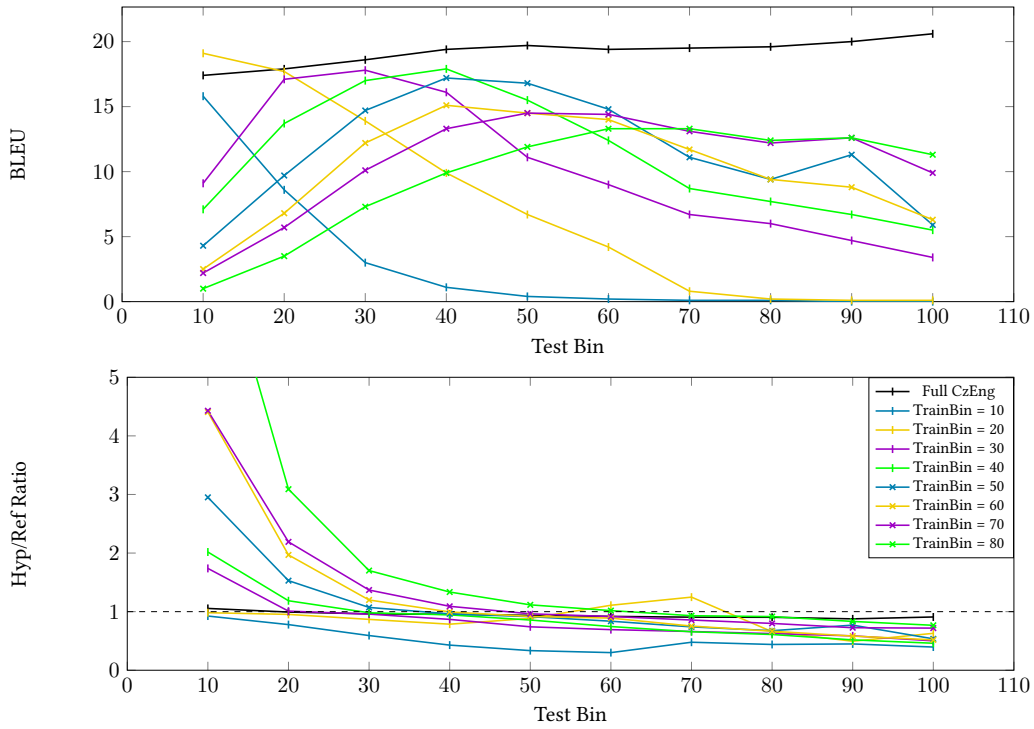

Figure 4.5: **Top**: Varying performance of Transformers on test data trained only on the data from a specific source-side length bin (various lines) when evaluated on a specific test bin (x-axis). BLEU scores are not directly comparable across different test sets (i.e. horizontally). **Bottom**: Average ratio between a hypothesis and reference. Dashed line indicates a ratio of 1.0. We reproduce the original figures from Variš and Bojar (2021).

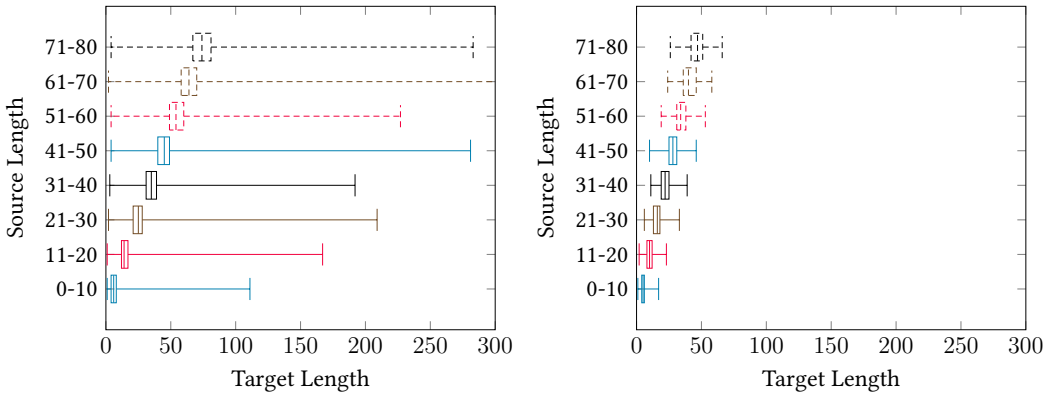

Figure 4.6: Distribution of lengths of target-side references within the training (**left**) and validation (**right**) datasets after splitting them into source-side length bin. Both figures have an identical x-axis scaling for better comparison. The long whiskers of the training bin length distributions are a result of noise in CzEng 2.0 training corpus. We reproduce the original figures from Variš and Bojar (2021).

resulting in sensible translation with less important source-side information being dropped (ignoring that the *company headquarters* is located *in Žižkov*). However, further investigation is required to confirm/disprove the consistency of the summarization abilities of the system.

Source (30-bin)	The company does not collect its mail and it has closed its official headquarters in Žižkov more than six years ago.
Hyp1 (10-bin) *Hyp1 (gloss)*	Společnost nesbírá poštu a zavřel oficiální sídlo. *The company does not gather mail and closed official headquarters.*
Hyp2 (30-bin) *Hyp2 (gloss)*	Společnost neshromažďuje poštu a již před více než šesti lety zavřela své oficiální sídlo v Žižkově. *The company does not collect mail and more than six years ago closed its official headquarters in Žižkov.*
Hyp3 (60-bin) *Hyp3 (gloss)*	Společnost nevybírá poštu a uzavřela své oficiální sídlo v Žižkově více než šest let ago. v Žižkově. Společnost neshromažďuje poštu a uzavírá oficiální ústředí v Žižkově více než šest let agr. o. *The company does not pick up mail and closed up its official its official headquarters in Žižkov more than six years ago. in Žižkov. The company does not collect mail and closes up official headquarters in Žižkov more than six years agr. o.*
Reference (30-bin) *Ref (gloss)*	Nepřebírá poštu a oficiální sídlo na Žižkově zrušila před více než šesti lety. *(The company) does not collect mail and official headquarters in Žižkov closed up more than six years ago.*
Source (30-bin)	The perpetrators ended up in custody, said Marie Štrbáková, the spokeswoman of Olomouc police.
Hyp1 (10-bin) *Hyp1 (gloss)*	Mluvila s ní Marie Štrkováková *Talked to her, Marie Štrkováková*
Hyp2 (30-bin) *Hyp2 (gloss)*	Pachatelé skončili ve vazbě, řekla Marie Štrbákováová, mluvčí Olomouckého policie. *The perpetrators ended up in custody, said Marie Štrbákováová, the spokeswoman of Olomouc police.*
Hyp3 (60-bin) *Hyp3 (gloss)*	Uchazeči skoncovali v úschově, "uvedla Marie Štrbákováová, mluvčí Olomoucké policie, která se stala mluvčí Olomouckého vojska, a to v úschově. *The candidates ended up in storage, "introduced Marie Štrbákováová, the spokeswoman of Olomouc police, which became the spokeswoman of Olomouc army, and in storage.*
Ref (30-bin) *Ref (gloss)*	Pachatelé skončili ve vazbě, informovala olomoucká policejní mluvčí Marie Štrbáková. *The perpetrators ended up in custody, informed Olomouc police spokeswoman Marie Štrbáková.*

Figure 4.7: Example translations from systems trained on specific target-length-restricted datasets. Both examples demonstrate the over- and under-generation of systems trained on datasets containing longer (60-bin) and shorter (10-bin) sentences when applied to inputs with the length of reference translation different from the training data (30-bin). We provide rough, *word-for-word* translations of the produced outputs (in *italics*) with color highlighting of the selected phrases and their corresponding English translation for better comprehension. The underline highlights grammatical errors or mistranslations in the output. The original translation examples were reproduced from (Variš and Bojar, 2021).

Based on the overfitting results, we decided to investigate, what is the main reason behind such model behavior. By design, the decoding algorithm inside Transformer generates output tokens recurrently one-after-another until the stopping criterion is met. This stopping criterion is reading a special end-of-sequence (EOS; marked as `<EOS>`) token as the input or reaching the maximum output length threshold. Therefore, the length of the output sequence is strongly dependent on the generation of this special token.

Due to the negative log-likelihood (NLL) training criterion and the fact that only a single reference translation is provided during training, the output probability of the EOS token becomes close to zero at positions where the EOS was not seen during training. On the other hand, the output of each token is not conditioned only on the current decoding position but also on the previously generated output. If the model can generalize well enough, it should, for example, be able to abstract a rule that after punctuation, such as full-stop, exclamation mark, or question mark, there should be a rather high output probability of the EOS token. Besides, there can be other indicators of a probable or improbable EOS, for example, a verb in verb-final languages or (un)finished valency of verbs, nouns, etc. (Panevová, 1994; Panevová, 1974), just to name a few.

We compared the EOS output probability dynamics between the baseline (Full CzEng) model and the *in-domain* 50-bin model. We translated the MT testset while also printing the output probability of the EOS token at each timestep. Figure 4.8 displays the comparison between the average output probability of the EOS token at each decoding position. As expected, the average output probability in the CzEng model (Figure 4.8, top) is fairly consistent regardless of the position. The in-domain model in contrast pushes the EOS probability close to zero when outside of the EOS *training positions*. Interestingly, the model is more inclined towards generating EOS earlier when translating sentences with shorter references (Figure 4.8, middle) and much later when translating sentences with longer references (Figure 4.8, bottom).

The results suggest that the EOS generation is strongly conditioned on the decoding position rather than the previous context. This is further supported by the case study examples in Figure 4.7 (60-bin).

Lastly, we investigated whether the observed length-based overfitting behavior can be mitigated via dataset augmentation. First, similarly to Kondo et al. (2021), we compared a regular 60-bin model with models trained on synthetic 60-bin data created by concatenation of shorter datasets, namely 10-, 20- and 30-bin. We shuffled each dataset and created the synthetic 60-bin data by concatenating every 6, 3, and 2 examples respectively to create synthetic examples of target length between 51–60 tokens.

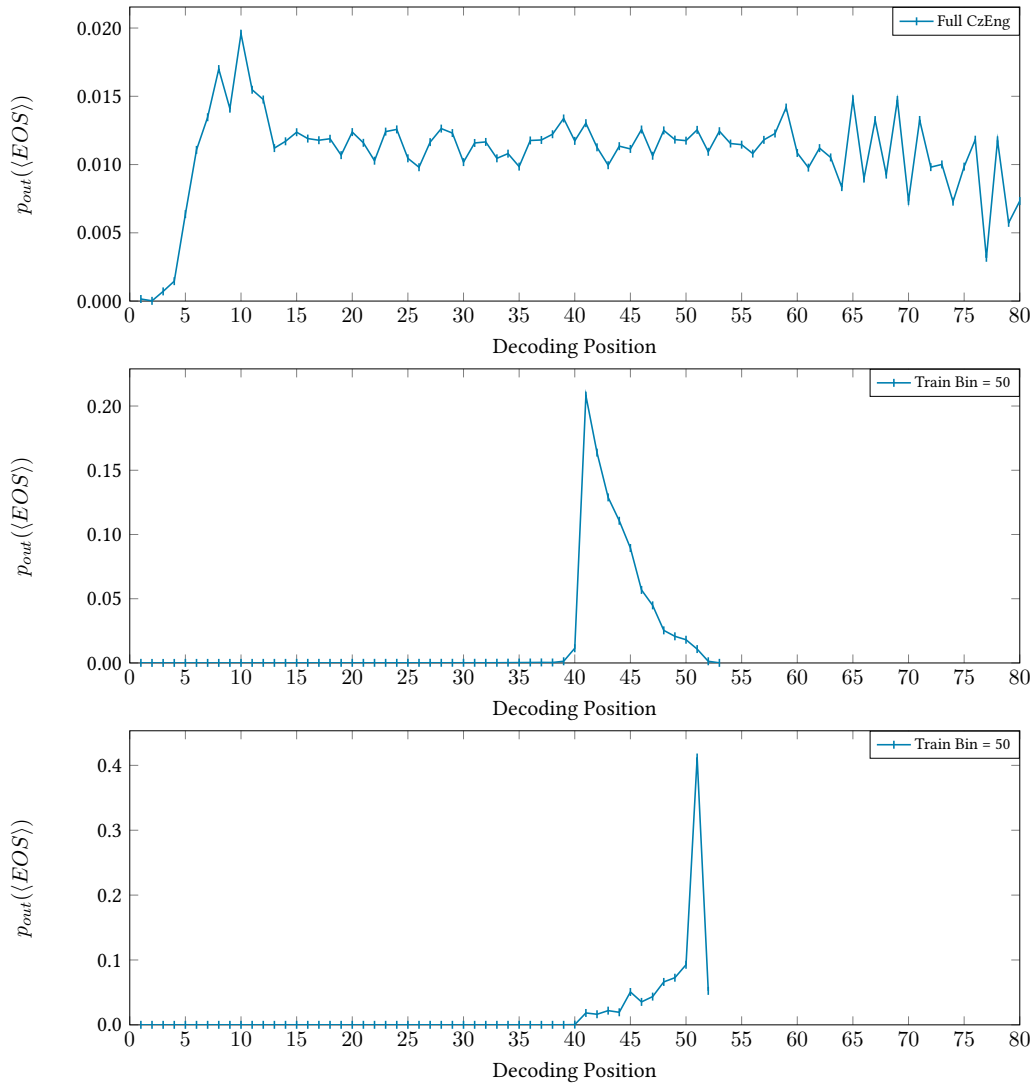

Figure 4.8: Emission probabilities of the <EOS> token at various decoding positions. **Top**: Average output probability of a model trained on the whole CzEng dataset, averaged over the full testset. **Middle**: Average output probability of a model trained on the 50-bin dataset, averaged over the shorter sentences from the testset. **Bottom**: 50-bin model, averaged over the longer sentences from the testset.

Figure 4.9 shows the performance of these systems evaluated similarly to the original length-based overfitting experiments. Even though most of the time the concatenated examples in the synthetic data are not related in any way, the concatenation itself (with the absence of the genuine 60-bin data) is enough for the models to reach comparable performance. The only exception is the synthetic 10-bin model; the drop in its performance is most likely due to not being able to capture dependencies spanning over more than 10 tokens which play an important role in translating sentences from the higher-bin testsets.

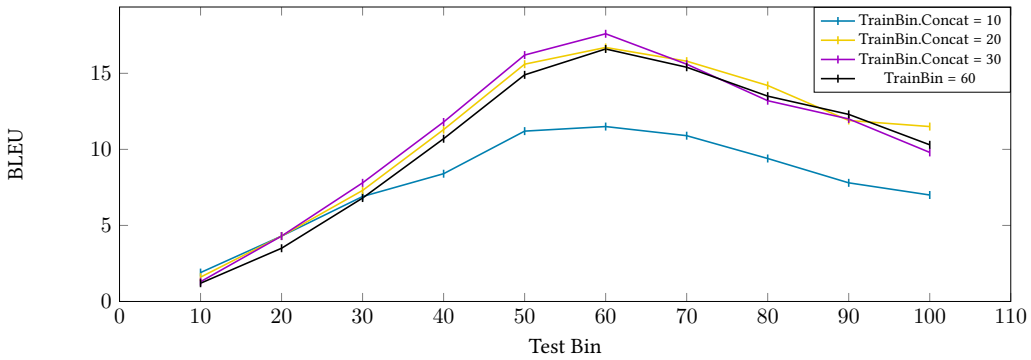

Figure 4.9: Comparison of the performance of a model trained on genuine data from the 60-bin dataset with models trained on synthetic 60-bin datasets created by concatenation of 10-, 20- and 30-bin sentences respectively. We reproduce the original figure from Variš and Bojar (2021).

Second, we took a closer look at the model behavior when it is presented with training data from a specific length span (11–50 tokens) containing a gap (i.e. not including examples from the 21–40 span). We think that a model that generalizes well, even though it cannot correctly model longer and shorter sentences, should be at least able to properly cover this *gap* that is not present during training.

Figure 4.10 shows a comparison of the performance of the model trained on the combination of the 20-bin and 50-bin datasets (20+50-bin) with models trained using one or the other in isolation. The performance of the 20+50-bin model is outperforming the models trained only on the single-bin datasets, but, the combination of two datasets results in a larger training sample with more diversity. Even the clear improvement on the 30- and 40-bin testsets does not necessarily guarantee better length-based generalization.

To get more insight into the 20+50-bin model behavior, we again inspected EOS emission probabilities. Figure 4.11 shows that the original length-overfitting issue does not get solved by the dataset combination, not even for the *gap* test examples. The areas with higher EOS emission probabilities are still concentrated near sequence positions where the symbol was frequently present during training. This further supports the hypothesis of strong conditioning of the EOS generation on the absolute decoding position. Furthermore, if we only look at EOS emission probabilities when translating sentences with references of length 31–50 (30- and 40-bin), it is clear that the MT system prefers over-generating instead of stopping the decoding too soon. We conclude that such behavior could be a result of an internally learned coverage mechanism, however, we leave the further investigation to future work.

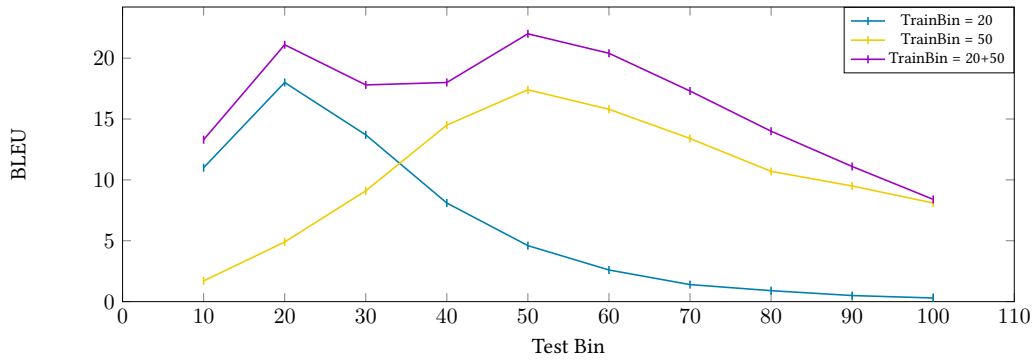

Figure 4.10: Comparison of the performance of a system trained on a combination of the 20- and 50-bin data (20+50) with systems that were trained only on a 20-bin or 50-bin dataset.

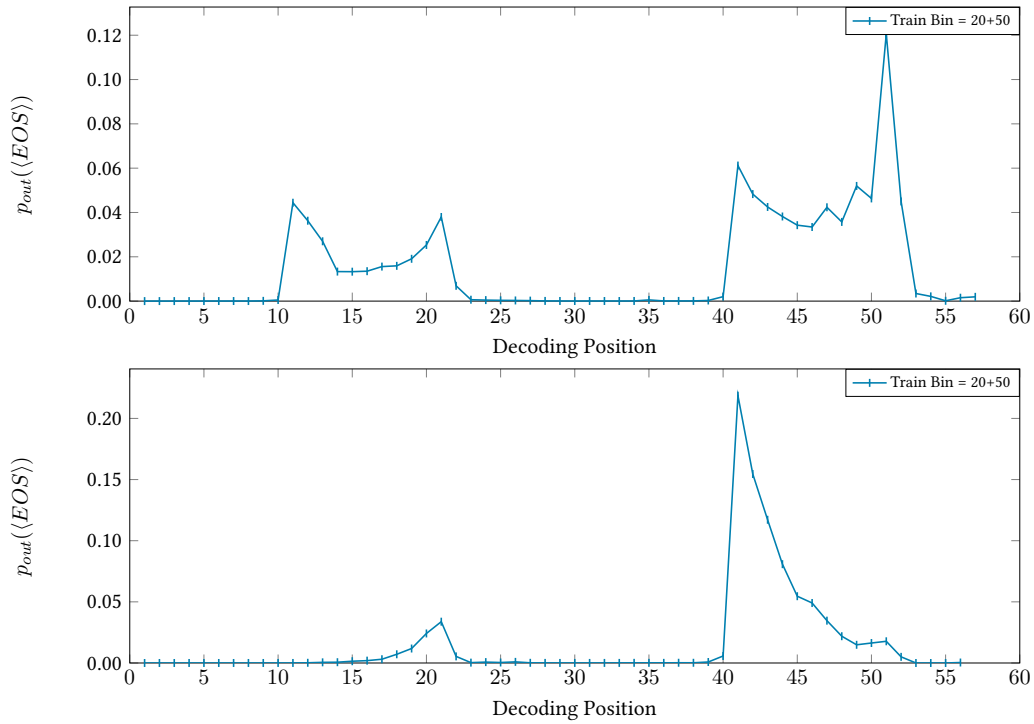

Figure 4.11: <EOS> emission probabilities of the system trained on the combination of the 20- and 50-bi dataset during decoding of the testset. Each position contains emission probabilities averaged across the dataset. **Top**: Average over the whole testset. **Bottom**: Average over the combination of 30- and 40-bin testset.

4.2 Exploiting the Word Distribution Similarities

We demonstrated a strong tendency of vanilla Transformers to overfit with respect to the target-side reference length distribution in the training data. The follow-up experiments will investigate whether Transformers exploit other dataset similarities when optimized using the available training data.

Human ability to translate is not only influenced by their previous experience in translation; humans are also able to effectively separate the underlying structure of the language and correctly apply such structural principles when facing language (i.e. a body of text) with novelties, i.e. previously unseen words or phrases. Humans can still understand the meaning of new words or phrases using various context clues (Al-Jamal, 2018; Hiebert and Kamil, 2005) A machine that can generalize well, should be able to mimic such behavior to a certain extent.

The previous research on image captioning has shown a strong tendency of DL models to exploit image representation similarities between the training/test data leading to an overestimation of the generalization ability of these models (Madhyastha et al., 2018). Similarly, we hypothesize that current Transformers, to some extent, exploit similarities between the current MT validation data and the available training datasets. To validate this hypothesis, we set up an experiment where we try to adversarially remove training examples that are most similar to the available validation data using similarity metrics related to vocabulary distributions. A system that can generalize well should still be able to learn from the remaining training examples without a significant decrease in translation performance.

4.2.1 Experiments

We use CzEng 2.0 (Kocmi et al., 2020) training data in the following experiments. We remove all data outside of the news-related domain, keeping only training examples from the *news* and *news-commentary* subset of the data, resulting in slightly more than 300k training sentence pairs. In practice, some form of overfitting can be desired, for example, in domain adaptation. For this reason, we focus only on the sole news-related domain to avoid model underfitting by filtering out distributionally similar, in-domain examples (keeping only the less similar, out-of-domain data), avoiding the so-called *domain shift*. We use WMT20 `newstest13-20` for model evaluation, using the following splits: `newstest17-20` for the final evaluation, `newstest15` and `newstest16` for training-time validation (and early-stopping) and `newstest13` and `newstest14` for the purposes of removing examples from the training dataset that have high vocabulary distribution similarity with the evaluation data. We use the same tokenization and BPE preprocessing as in the MT experiments from Section 4.1.1. The models trained are based on the models from Appendix A.3

To investigate the effects of reducing distributional similarities between the training and validation datasets, we propose dataset filtering based on the following two methods: n-gram language model and term frequency-inverse document frequency (TF-IDF) cosine similarity. Both methods were applied to the corpora after the BPE-tokenization. Similarly, the subsequent analysis was performed on the BPE-

tokenized corpora. The first method trains the n-gram LM with smoothing (Chen and Goodman, 1996) using the small holdout data and uses it to compute the likelihood of the training sentences with regards to the LM. We use KenLM[13] (Heafield, 2011) implementation for its fast optimization of decoding using CPU. Then, we remove the sentences in the order from the most likely to the least likely.

The second method creates TF-IDF representations of the sentences in the holdout data, computing the IDF portion of the formula by considering each holdout sentence as a separate *document*. The fitted TF-IDF model is later used to create representations of the training sentences and the similarity of each training sentence with the holdout data is computed by taking the maximum similarity that a given training sentence has with the individual holdout sentences. Again, we use the resulting scores to remove the training sentences from the most similar to the least similar. For control, we compare these methods with a simple random filtering scheme, ordering the sentences randomly before the removal. Due to the nature of the MT data (sentence pairs), we compare similarity filtering based on both source- and target-side dataset similarities.

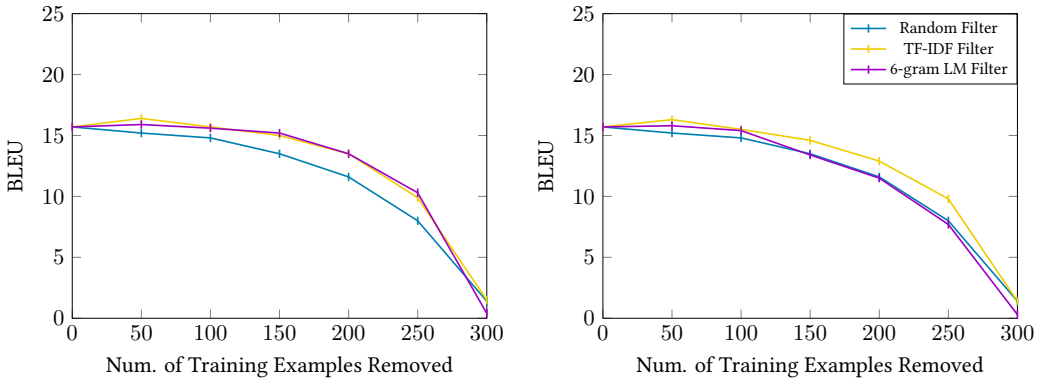

Figure 4.12: Model performance degradation after removing the predetermined number of examples from the training data, evaluated using the `newstest17-20` testset. **Left**: Dataset filtering based on the source-side similarity scores. **Right**: Filtering based on the target-side similarity.

Figure 4.12 shows the performance degradation with respect to the proposed dataset filtering schemes. Contrary to our hypothesis, systematic removal of training examples based on either the LM or the TF-IDF similarity scores does not lead to a larger drop in BLEU than after randomly removing a similar number of training examples. More surprisingly, both the source- and target-side TF-IDF filtering initially even slightly improve the model performance (50k removed examples), suggesting

[13] `https://kheafield.com/code/kenlm/`

that the TF-IDF filtering can have a noise reduction potential. However, we leave the analysis of this aspect of the filtering to future work. The only instance where the scoring-based sentence removal leads to a noticeable drop, although still performing similarly to the random removal, is the target-side LM-based filtering.

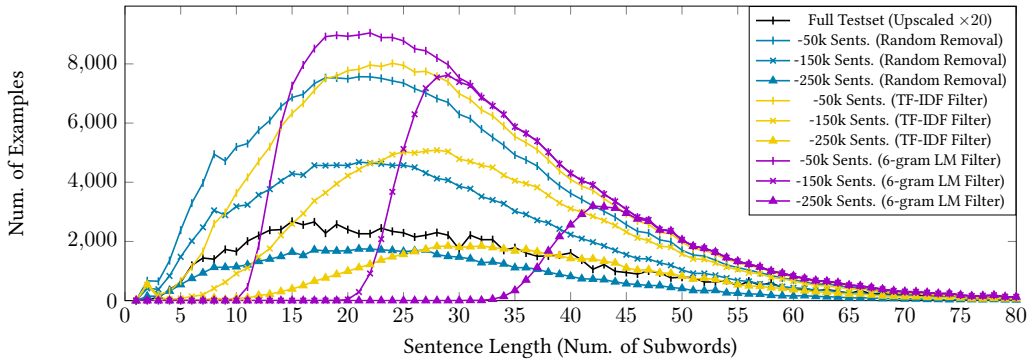

Figure 4.13: Training dataset target-side sentence length distribution after removing a specific number of the training examples with respect to various filtering methods. The upscaled length distribution of the testset is provided for comparison.

To understand the effects of the filtering methods we looked at two aspects of the resulting training datasets: sentence length distribution and vocabulary distribution. Figure 4.13 shows the target-side sentence length distributions of the training dataset created by various filtering methods. While the random and TF-IDF-based removal leads to datasets that have length distributions similar to the test data, the LM-based removal method prefers to remove shorter sentences first, possibly due to shorter sentences having a higher likelihood given our LM. Combined with the observations in Section 4.1.1, this most likely leads to the model overfitting to longer sentences, resulting in a drop in BLEU drop due to over-generation when translating the test data.

The BLEU difference between the random and 6-gram LM filter is much smaller when applying target-side filtering, implying higher importance of target side-sentence similarity than that of the source-side. Such a result could explain the effectiveness of the backtranslation (Sennrich et al., 2016a) data enhancement approach – the mistranslations on the source side which lead to a different source-side sentence distribution compared to the genuine parallel data do not negatively affect the performance of the resulting model as long as the target-side sentences are from the genuine distribution. Still, the drop is not more significant than in models trained on datasets with randomly removed training examples. Although the removal based on the source-side filtering also leads to the removal of shorter training examples, its effects are not as strong as with the target-side LM removal. This is mostly due to the varying ratio between the source and target-side sentences in the data as we demonstrated earlier in this chapter.

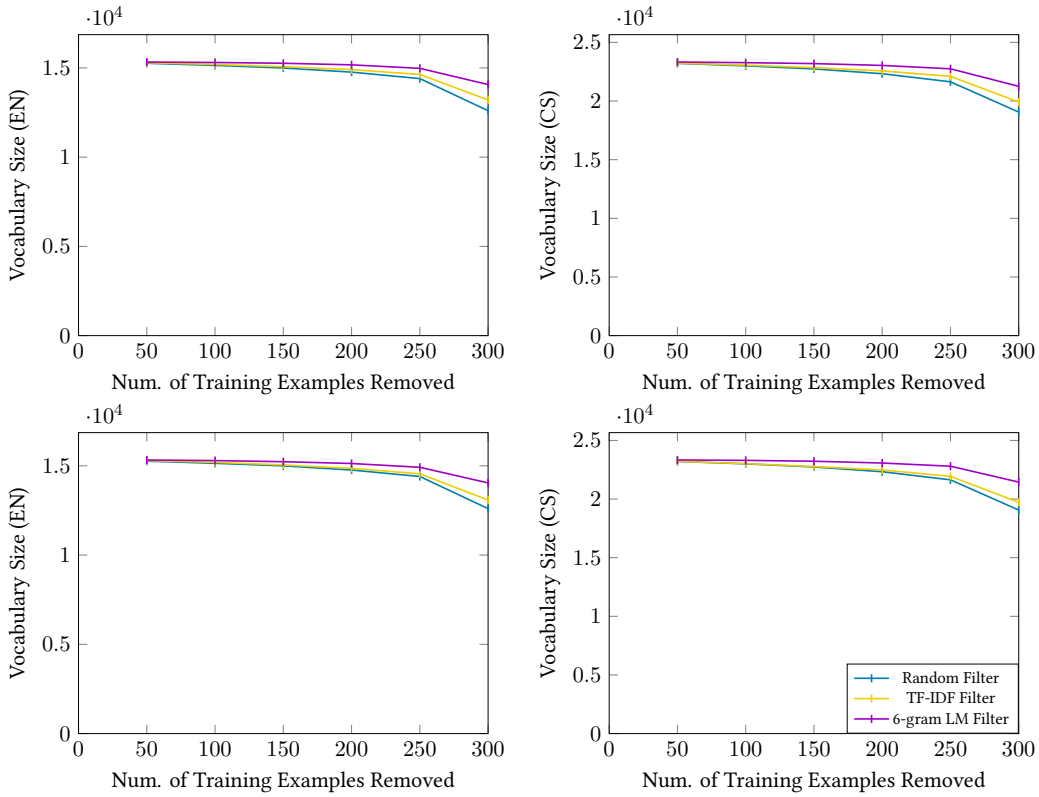

Figure 4.14: Vocabulary sizes of the resulting training corpora. **Top**: Dataset filtering based on the source-side similarity scores. **Bottom**: Filtering based on the target-side similarity.

Next, we inspected how the filtering methods affect the vocabulary distributions of the resulting training data. Figure 4.14 shows that the trends are similar regardless of the filtered side (source-side, target-side filtering) or the language choice: both random and TF-IDF methods lead to a higher vocabulary reduction than the LM-based filter. This suggests that the sole vocabulary reduction is not the reason behind the BLEU score difference between the TF-IDF and random training example removal.

To get additional insight into the differences between the filtering methods, we also compared the KL differences between the unigram vocabulary distributions of the filtered training data and the vocabulary distribution of the testset. For each dataset, we computed the probability of each vocabulary entry based on its frequency compared to the overall number of tokens within the dataset resulting in the unigram distribution for the given dataset. Figure 4.15 displays the KL divergence between the unigram distributions of the various training datasets and the testset. Contrary to the random and LM filtering, TF-IDF similarity removal leads to a significantly different unigram distribution, most apparent in the distribution of the 100 most frequent tokens in our test data. This contradicts our original hypothesis: while the TF-IDF filtering leads to a less similar dataset (in the terms of unigram token distribution), it

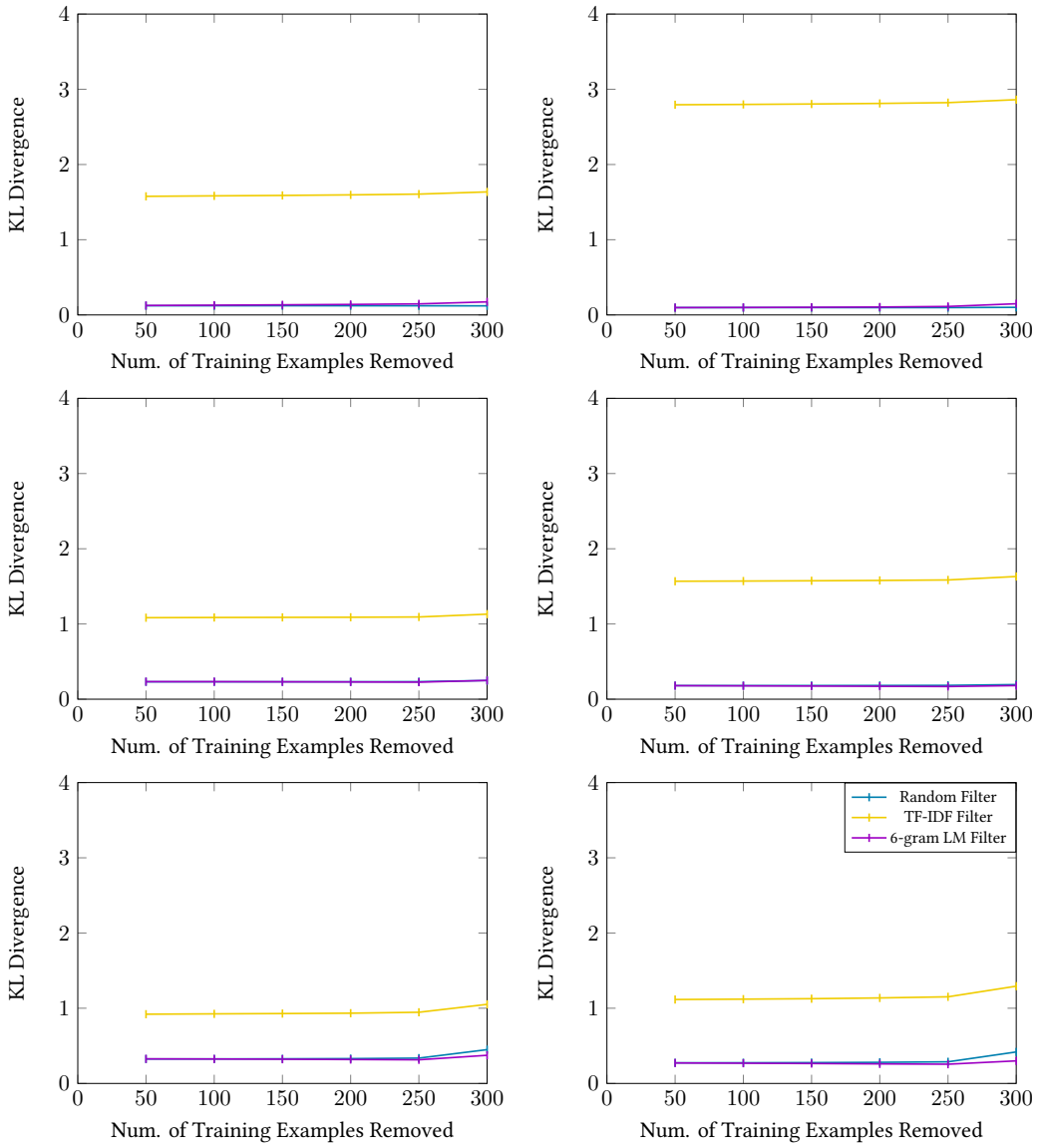

Figure 4.15: Kullback–Leibler (KL) divergence between the unigram distribution in the filtered training corpora and the test dataset computed with respect to the top-100 most frequent words (**top**), top-1000 most frequent words (**middle**) and top-10000 most frequent words (**bottom**) in the test dataset, with probabilities normalized with respect to the given vocabulary subsample. **Left**: English source-side, **right**: Czech target-side.

does not lead to a higher performance drop (and even a slight improvement initially, based on the results in Figure 4.12). Thus, we conclude that Transformers do not exploit the similarities between the training/test data unigram token distributions to boost their performance.

4.3 Rare Word Transcription

First iterations of NMT systems (Sutskever et al., 2014; Bahdanau et al., 2014) were often limited by their fixed-size vocabulary, resulting in poor performance when translating rare words. This problem was effectively tackled by introducing subword-tokenization, such as byte-pair encoding[14] (BPE, Sennrich et al., 2016b) or SentencePiece.[15] The introduction of subwords improved the translation (and transliteration) of rare words, namely named entities, loanwords, and morphologically complex words (Sennrich et al., 2016b). Usually, these words can be translated (even by a human translator) without prior knowledge of the word by splitting the word into smaller units (subwords) and applying simple rules. For example, in the case of named entity translation, the word can be either transliterated (if the source and target language alphabet do not match) or most of the time just copied (while applying the target language rules for flection).

The following section explores the copy behavior applied during named-entity transcription and how well Transformers generalize with respect to this ability in the context of NMT. We hypothesize that this behavior is not learned in a general form but instead reflects the nature (e.g. subword length) of the copied words in the training dataset leading to the Transformer overfitting to the training instances.

4.3.1 Experiments

Similarly to the previous section, we use the news-related subset of CzEng 2.0 for training and the newstest validation corpora for evaluation. We also apply the same preprocessing pipeline on each dataset.

We derive the BPE merge rules using the whole training dataset (both English and Czech sides), setting a limit of 30k merges. Next, we create thresholded training datasets by removing training sentence pairs that contain tokens of a subword length that exceeds a given threshold. Again, we create separate training dataset versions using both source- and target-side filtering. This should guarantee that a model trained on a given dataset does not encounter tokens with subword lengths higher than the given threshold. We aim to measure how much this later affects the model's ability to transcribe named entities of various lengths during test time.

[14] `https://github.com/rsennrich/subword-nmt`
[15] `https://github.com/google/sentencepiece`

Orig. (En)	According to an architect, its regional plan resembles **Manhattan** in New York.
Orig. (Cs)	Územním plánem připomíná newyorský **Manhattan**, tvrdí architekt.
Repl. (En)	According to an architect, its regional plan resembles **Toepwgzhem** in New York.
Repl. (Cs)	Územním plánem připomíná newyorský **Toepwgzhem**, tvrdí architekt.

Figure 4.16: Example of the named-entity identification and replacement in the testset. The replaced token is in **bold**.

To measure the named entity transcription (copy) accuracy, we create the following variations of the final evaluation dataset: we identify named entities in the dataset using NameTag[16] (Straková et al., 2014) and collect all test set instances that contain a single named entity that has an identical surface form in both source- and target-side of the sentence pair. We consider these named entity translations to be a result of the copy operation. Next, we replace each found named entity with a randomly generated string with a capital first letter (emulating a foreign named entity) on both source- and target-side. Figure 4.16 illustrates the testset named-entity replacement.

We create multiple testset derivations, each containing only named entity replacement of a specific character length. We use these datasets to measure the model's accuracy to produce the target-side surface form of a named entity present on the source-side.[17]

We compare the baseline system trained on the whole news-domain with systems with varying BPE thresholds. For a fair comparison and to avoid test-time out-of-vocabulary (OOV), each model is initialized with the same vocabulary extracted from the full, BPE tokenized, news-domain corpus. The model hyper-parameters are described in Appendix A.3.

Figure 4.17 shows the accuracy of the randomly generated named entity copying with respect to the varying character and subword length of the copied named entities. As the threshold indicating the maximum subword length of tokens in our training data decreases, the accuracy of the named-entity copying decrease too. This effect is more noticeable as the subword length of the copied named-entities increases. A significant deterioration starts at thresholds lower than 4 – this might be due to a more significant reduction of the training data resulting from the filtering. Surprisingly, a larger drop in accuracy of copying longer named-entities occurs after the source-side filtering of the corpora. This is opposite to our previous observation that

[16] `https://ufal.mff.cuni.cz/nametag/1`

[17] Note that we do not directly measure whether the named entity is produced in a correct context. Furthermore, we choose named entities that have an identical source- and target-side surface forms in our test data to avoid measuring the system's ability to correctly flect the foreign named entities, i.e. when translating to Czech. We are aware that this approach does not completely remove the grammatical necessity of the flection.

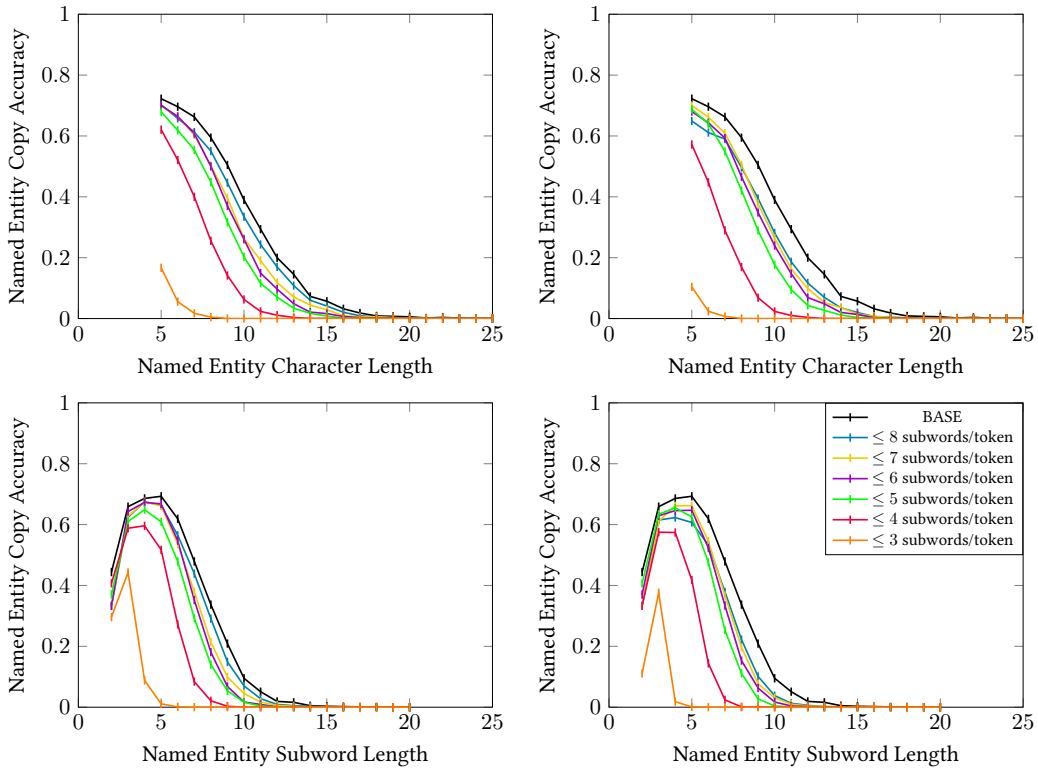

Figure 4.17: Named Entity copy accuracy with respect to the character length (**top**) and subword length (**bottom**) of a given named entity. **Left**: Model performance when trained on datasets with a source- side subword length threshold. **Right**: Models trained on datasets with thresholded target-side.

the models tend to overfit mostly to the target-side data. The target-side overfitting can still be noticed in the accuracy of copying shorter named-entities. The models with lower thresholds (but larger than 4) achieve better accuracy than the more "general" models with higher filter thresholds.[18]

To confirm that the subword-length threshold affects mostly the copying ability of the Transformers we also measured their performance on the dataset with the specific subword length named-entity replacements (Figure 4.18). Based on the results, the removal of training examples that contain tokens with a length above the threshold does not significantly impact the overall model performance. Only after a larger reduction of the threshold (less than 4) do the effects start impacting the resulting BLEU scores.

[18]Note that the low accuracy on the testset with named-entities with BPE length 2 is most likely due to a small dataset sample. The samples were generated based on the character length of the named-entities, tokenized using BPE, and clustered based on their BPE length for the evaluation in Figure 4.17 (bottom).

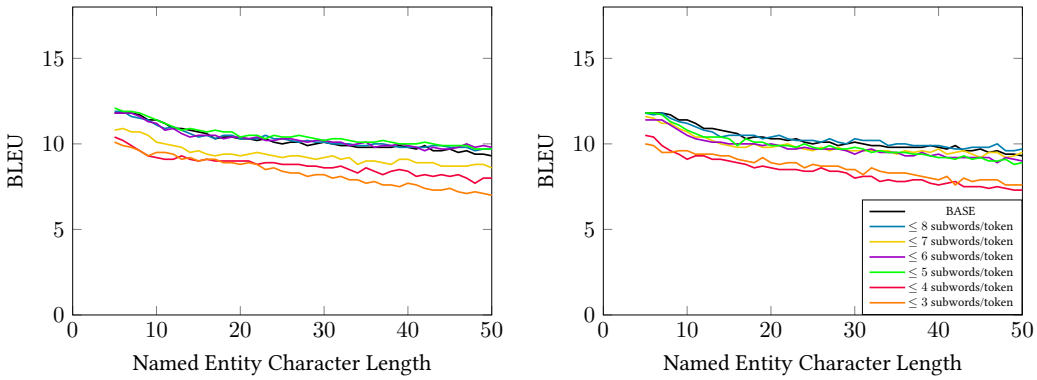

Figure 4.18: BLEU performance of models trained on datasets with varying subword-length threshold with respect to the character length of the generated named entity replacement in the test data.

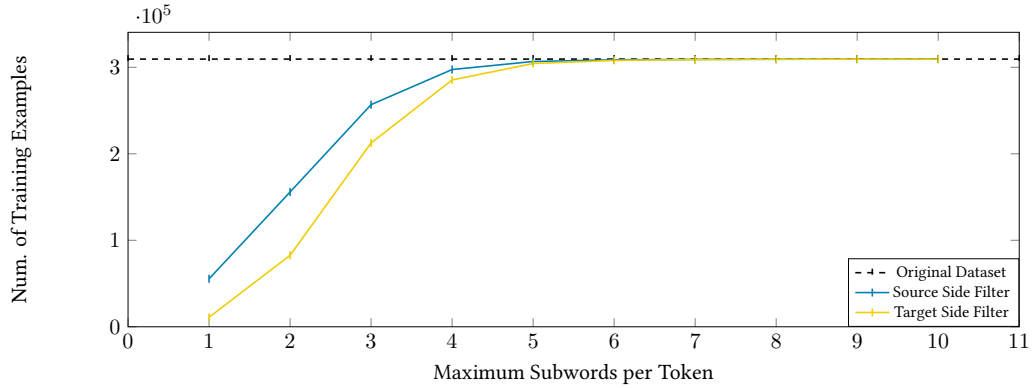

Figure 4.19: Training dataset sizes after applying various subword length threshold filters.

Since the main caveat of the comparison of our system is the reduction of the size of the training data through the removal of the training examples that do not satisfy the threshold criterion. However, as demonstrated by Figure 4.19, the significant training data reduction begins to be apparent only when the subword length threshold becomes lower than 4 subwords. This explains the large drop in both BLEU and copy accuracy of the models trained on the resulting smaller datasets. Still, it is unlikely that the size of the training data had a significant effect on the models. The models trained on data with the higher subword length threshold had only several hundred to thousands of examples removed.

4.4 Conclusions

The experiments in this chapter explored various limitations of the original Transformer NMT architecture. The results helped us answer **Research Question 1:** *What is the extent of the generalization ability of the current Transformer models?*, proposed in the thesis introduction. We decided to answer this question by breaking it down into three sub-questions. What follows is a revisit to these sub-questions and our answers based on the presented empirical results of our experiments.

RQ1.1 *Are long-range dependencies the only reason behind the performance drop when translating long sequences?*

Experiments in Section 4.1 demonstrated that the original Transformer fails to translate very long sequences but can also suffer a performance drop whenever it is lacking data from specific target-side length ranges, shorter or longer. We identified that the main reason behind this behavior is the tendency of the Transformer to generate sentence lengths similar to the ones seen during training. This is further confirmed by the analysis of the EOS token emission probabilities which shows that the Transformer is unlikely to terminate sentence generation at positions where EOS symbols were not seen during training. Lastly, we show that overfitting can be mitigated by enhancing the training dataset using synthetic long sequences created by the concatenation of shorter training examples.

RQ1.2 *Are the current NLP approaches to dataset splitting sufficient to properly measure the generalization of sequence-to-sequence models?*

Contrary to our original hypothesis, the removal of training examples based on the vocabulary distribution similarity with the holdout data (both TF-IDF and n-gram based) did not lead to a bigger decrease in the model performance compared to the random removal baseline, suggesting that Transformers do not overfit to the vocabulary distribution in the training dataset.

RQ1.3 *Is the ability to copy unseen words related to the subword length of tokens seen during training?*

We demonstrated that reducing the maximum subword length of tokens present in our training data without a significant reduction of the training dataset size can reduce the ability of the Transformer to translate rare words by copying. Due to the copy operation not always being the correct translation rule for rare words, we restricted the experiment only to the named entities that had identical surface forms on both the source and target sides in the test dataset. Even though the accuracy of

the rare word copying reflects the subword length restriction on the training dataset, a drop in BLEU scores becomes significant only when the restriction results in a removal of a larger portion of the training dataset. This suggests that the Transformer ability to copy rare words is related to the subword lengths of tokens present in the training data.

5

Incremental Learning and Catastrophic Forgetting

The artificial general intelligence (AGI) that aims to reflect human-like behavior does not only require to be highly versatile, i.e. by being able to tackle a variety of tasks and generalize well when facing new instances of a previous problem, it also needs to be able to further improve given new information while building upon its previous knowledge (Lake et al., 2017; Biesialska et al., 2020). Although state-of-the-art (SoTA) deep learning can tackle learning multiple tasks at once quite well (Zhang and Yang, 2017; Chen et al., 2021b), the models often struggle when trying to learn to solve these tasks sequentially. In literature, this is mainly attributed to the phenomena of catastrophic forgetting (CF) or catastrophic interference (CI, French, 1999; McCloskey and Cohen, 1989). This occurs when a neural network trained to solve one task (Task A) is later optimized for solving another task (Task B), as illustrated by Figure 5.1. Without any form of constraint on the training algorithm, the network weights optimized for Task A are usually overwritten (forgotten) when searching for the optimal solution for Task B. This behavior can be observed even when the network has the capacity to learn both tasks, e.g. through joint objective learning (de Masson d'Autume et al., 2019a). Often, there can be several network parameter configurations that result in a low error for each given task (Hecht-Nielsen, 1992; Sussmann, 1992).

Learning a task requires searching for a configuration of network weights which results in low-error performance on that particular task, usually defined by a loss function relevant to the task. In general, many weight configurations can result in very similar model performance (Hecht-Nielsen, 1992; Sussmann, 1992). Learning to solve multiple tasks, therefore, requires finding an intersection between the low-error weight configurations for both Task A and Task B. In incremental learning,

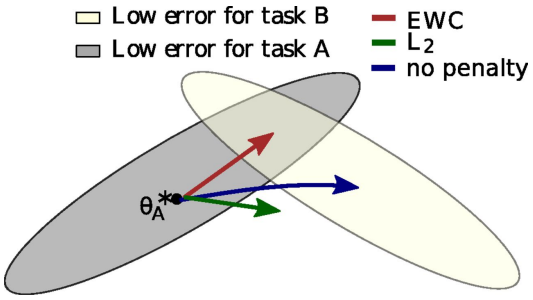

Figure 5.1: Reason behind catastrophic forgetting, proposed by Kirkpatrick et al. (2017). When a model is initialized with weights that achieve low error for Task A and fine-tuned for Task B, lack of explicit regularization can lead to a solution that has low error only for Task B even though a shared low error solution for both tasks exists.

this usually means shifting the original solution $\boldsymbol{\theta}_A$ for task A to a low-error area of Task B.[1] If we assume no access to the original data for Task A, optimization for Task B without any training constraints can result in completely leaving/avoiding the parameter subspace with low error values for Task A. Some form of regularization during the network fine-tuning is therefore required.

While joint task learning (or multi-task learning, Luong et al., 2016; Hashimoto et al., 2017) does not need to deal with CF, it requires that the data for both Task A and Task B to be available at the same time; a requirement that can sometimes be hard to satisfy in practice. Another option is storing network weights from the previous task separately and learning weight configuration for the new task by fine-tuning the previous model and having a separate *expert* model for each task. This, however, can lead to a weaker generalization due to each model being overfitted to its respective task. In addition, the storage requirements needed to keep a separate weight configuration for each task can be limiting. Hence, we are interested in methods that aim to restrict the network in a way that it does not completely overwrite its knowledge about the previously learned tasks.[2]

Multiple methods for avoiding or mitigating CF have been proposed throughout the years based either on the explicit model regularization (Hinton and Plaut, 1987; Kirkpatrick et al., 2017; Zenke et al., 2017), rehearsal (Robins, 1995; Draelos et al., 2017) or ensembles (Polikar et al., 2001; Dai et al., 2007).

[1] Assuming that the original solution for Task A is not already optimal also for Task B.

[2] By referring to the knowledge we mostly mean information stored in the network weight configuration.

In this thesis, we focus on regularization-based methods, mainly elastic weight consolidation (EWC, Kirkpatrick et al., 2017). While these methods were already established in the previous research, there has been only minimal study of their application and their limitations in tandem with Transformer network architecture and with sequence-to-sequence learning in general.

5.1 Elastic Weight Consolidation

Similarly to other connectionist models, Transformer models store the information about learned tasks in their trainable parameters. Generally, the information is distributed, meaning that knowledge learned about a single data point is represented by a subset of these parameters (McCloskey and Cohen, 1989). The distributed representations that are mainly a result of the applied learning algorithms contain properties that are claimed to be suitable for modeling human cognition, such as content-addressable memory and generalization (Hinton et al., 1990; McClelland et al., 1986; Pinker and Prince, 1988; Fodor and Pylyshyn, 1988; Lachter and Bever, 1988). They are, however, ale one of the major reasons behind the CF in incremental learning (IL, McCloskey and Cohen, 1989).

The goal of the learning algorithms is to find a set of network parameters that minimize a given loss (error) function with respect to the data available for respective tasks. However, some parameter configurations, while being optimal for one task, can be far from optimal when applied to the other tasks as illustrated by Figure 5.1. Thus, guiding the learning algorithm to stay close to a parameter subspace of solutions to the former tasks while learning new tasks poses a reasonable solution to avoiding CF.

Compared to neural networks, biological brains can learn and store new information in a sequential manner (Cichon and Gan, 2015). Kirkpatrick et al. (2017) suggest that the identification and consolidation of network weights with high *importance* with respect to previously learned tasks, and reducing the size of the updates to these weights can be a viable approach to reducing CF. Similarly to biological neurons, the ability to strengthen the retention of the important parameters, forcing the NN to update those that are irrelevant to the previous tasks could be crucial to prevent CF.

Regularization aimed at restricting updates to network parameters that are important for previously learned tasks should, in theory, motivate the network to make use of its remaining capacity. Besides regularization, such behavior can be also achieved by introducing constraints directly to the network parameters (Pasunuru and Bansal, 2019). Instead of updating the network parameters directly during the optimization for the subsequent tasks (Task B), the authors suggest learning a

parameter-shift $\boldsymbol{\psi_B}$ that together with the original task (Task A) parameters $\boldsymbol{\theta_A}$ can be used for computation of i-th model parameter for Task B, $\theta_{i,B} = \theta_{i,A} + \psi_{i,B}$. They report that by forcing block-sparsity on the Task A parameters $\boldsymbol{\theta_A}$ and parameter-shift $\boldsymbol{\psi_B}$ and making both parameter matrices orthogonal, the resulting parameters $\boldsymbol{\theta_B}$ will be both solutions for Task B while retaining knowledge about Task A.

Instead of forcing explicit constraints on the parameter matrices, Kirkpatrick et al. (2017) suggest deriving network parameter importance directly by framing incremental learning as a Bayesian inference problem. Given training datasets $\boldsymbol{D_A}$ and $\boldsymbol{D_B}$ representing tasks A and B, respectively, the subsequent optimization for the latter task after the optimization for the former one can be described by the following formula:[3]

$$\log p(\boldsymbol{\theta}|\boldsymbol{D}) = \log p(\boldsymbol{D_B}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}|\boldsymbol{D_A}) - \log p(\boldsymbol{D_B}) \tag{5.1}$$

where $\boldsymbol{D} = \boldsymbol{D_A} \cup \boldsymbol{D_B}$. The authors claim that if the likelihood term $\log p(\boldsymbol{D_B}|\boldsymbol{\theta})$ represents the negative of the loss function for optimization with respect to Task B, the prior $\log p(\boldsymbol{\theta}|\boldsymbol{D_A})$ should contain the information about the previously learned task A and consequently, the information about the importance of the network parameters regarding Task A.

In practice the conditional probability $\log p(\boldsymbol{\theta}|\boldsymbol{D_A})$ is intractable during the Task B optimization, therefore, Kirkpatrick et al. (2017) suggest estimating the prior probability distribution as a Gaussian distribution with the mean being the parameters $\boldsymbol{\theta^*_A}$ at the end of Task A optimization and variance estimated using a diagonal of the Fisher information matrix (FIM).[4,5] The properties of FIM (equivalency to the second derivative of the loss near the minimum and positive semi-definitiveness) allow the formulation of the following regularized loss function:

$$L(\boldsymbol{\theta}) = L_B(\boldsymbol{\theta}) + \frac{\lambda}{2} \sum_{i:\theta_i \in \boldsymbol{\theta}} F_{i,A}(\theta_i - \theta^*_{i,A})^2 \tag{5.2}$$

The regularization term penalizes updates to the parameter θ_i more when its importance F_i is higher.

[3]Huszár (2017) points out that the original authors wrongly assume that $\log p(\boldsymbol{D_B}|\boldsymbol{D_A}) = \log p(\boldsymbol{D_B})$ based on the mutual independence of the data samples from $\boldsymbol{D_A}$ and $\boldsymbol{D_B}$. Huszár (2017) states that the "dependence is induced by the Bayesian treatment of the problem: the tasks are no longer assumed independent, instead, they are exchangeable (conditionally independent given $\boldsymbol{\theta}$)".

[4]Inspired by the previous work on Laplace approximation (MacKay, 1992).

[5]In practice, we approximate FIM using *empirical Fisher* (Schraudolph, 2002a). Note that while the two matrices are not identical, if not stated otherwise, our use of the term FIM refers mainly to the empirical Fisher approximation.

Alternatively, instead of estimating the importance of individual network parameters with respect to previously learned tasks by examining the loss space curvature near the parameters at the end of the training, Zenke et al. (2017) suggest tracking the contribution of each parameter to the loss change throughout the whole training process. When making infinitesimal updates to the network parameters, the change in loss can be estimated by a sum over gradients of the individual network parameters (multiplied by the size of an update). Using path integral (PI, Feynman, 1948), this contribution can be expanded to the whole sequence of updates between the initial parameter values and their values at the end of the task. The suggested importance measure for each parameter is the sum of the products of the parameter gradient and its update tracked throughout the whole training. Although the results of their experiments show similar performance to the EWC-based regularization, the PI-based regularizer can be better suited for online learning scenarios thanks to the tracking of the importance throughout the whole training.

Besides a direct regularization of the network parameters, disabling updates to the original parameter values and learning task-specific layer activations can be a viable approach (Li and Hoiem, 2016; Rebuffi et al., 2016). Instead of storing a new set of parameters for each subsequent task, having a single set of the original task network parameters and only a set of optimized layer activations for the subsequent tasks can effectively help reduce storage requirements for each subsequent task. Still, similarly to fine-tuning, this approach requires a strong prior model for the subsequent task optimization (Li and Hoiem, 2016).

5.2 Weight Consolidation for Unsupervised Pretraining

Although originally designed to reduce CF in neural networks, our first set of experiments investigated whether EWC can be to counter overfitting in unsupervised pretraining scenarios. Unsupervised learning in the area of NMT most often refers to only using available monolingual data, possibly for both source and target language respectively without any explicit notion of word or sentence-level alignment (Artetxe et al., 2018; Lample et al., 2017). Even though the monolingual corpora provide useful information about the structure of both the source and target language on their own, there is no explicit information about mapping sentences from one language to sentences in the other.[6]

[6] Although it is not the focus of our work, it is important to mention that there are also unsupervised NMT approaches that can optimize a system for translation solely from the monolingual corpora (Lample et al., 2017; Conneau and Lample, 2019a; Artetxe et al., 2018).

The unsupervised pretraining thus focuses on the first training parts of the machine translation (MT) model, in our case encoder and decoder, to learn about the structure of source and target-side language respectively. Next, these pretrained *language models* are used in the subsequent training on the available parallel data, providing better initialization of the model parameters. We consider both the encoder and decoder as isolated language models due to the training objective that is used in the pretraining phase: both are optimized to learn the probability distribution over the token vocabulary given a specific context (can be either prefix, suffix, or a complete context in the case of masked language models). During the NMT optimization itself, the decoder is then trained to condition said output probability distribution not only on the output history itself but also on the source-side sentence.

The biggest motivation behind the use of additional monolingual data in MT optimization is their much larger availability compared to their less-resourced bilingual or multilingual counterparts. Thus, there were various approaches to incorporating the additional monolingual data into the training pipeline. The idea of using additional corpora for building MT systems originated in the classical statistical machine translation (SMT), including the most famous model of phrase-based machine translation (PBMT). SMT models were originally based on the noisy channel model (Shannon, 1948) which applies the Bayes rule on the modeled posterior $p(\mathbf{y}|\mathbf{x})$ to decompose the probability into a separate *translation* $p(\mathbf{x}|\mathbf{y})$ and *language model* $p(\mathbf{y})$. Later statistical models, based on maximum entropy estimation (MEE), while allowing higher freedom in adding additional statistical components (e.g. reordering models, Galley and Manning, 2008; Koehn et al., 2005), they still included dedicated target-side language models trained on monolingual corpora.

In the context of NMT, the current most popular method of using monolingual corpora is the creation of additional synthetic parallel data by automatically translating the source and target-side monolingual corpus (Sennrich et al., 2016a). The latter, also known as back-translation, became a staple data-augmentation method in contemporary NMT research. Even though the source-side sentences created by this method contain translation errors, these additional training data provide the NMT decoder with examples that help further refine its conditional language modeling abilities.

The unsupervised pretraining can provide the NMT model with a good initialization, though, its benefits are more noticeable in the unsupervised NMT scenarios (Baziotis et al., 2021; Lample et al., 2017). Possibly due to the tendency of neural models to overfit to the training data, parameter initialization in itself via pretraining might not be making use of the full potential of the available monolingual data. Ramachandran et al. (2017) show that in addition to parameter initialization, the original LM objectives used during NMT pretraining can be also used as auxiliary train-

ing objectives during the NMT training, taking a role of a regularizer. The results of their experiments show that LM objective regularization can lead to translation performance on par with the backtranslation techniques. On the other hand, it requires that the original monolingual corpora are also available during the subsequent NMT training.

Inspired by their regularization approach, we investigated whether the LM objective regularization can be replaced by a different regularizer, namely one based on the EWC. Our reasoning is following: during the pretraining phase, the model components (encoder, decoder) learn the structure of the languages in isolation, and this prior knowledge about the languages is used as a basis for the subsequent NMT training (by initialization or by including the auxiliary LM objectives). As described by Kirkpatrick et al. (2017), EWC can be used to include these LM priors in the NMT learning. In the original paper, they introduce the Bayes formula for a scenario, when a whole model is being optimized for two tasks incrementally. In the following section, we derive a formula for incorporating prior pretraining of individual parts of the Transformer architecture.

5.2.1 EWC Regularization of Submodules

The original formula (Equation 5.1) motivating EWC considers continual training of the whole network. The modular nature of the Transformer (and other sequence-to-sequence architectures) allows a separate pretraining of the encoder and decoder sub-networks. Let $\boldsymbol{D} = \boldsymbol{D_{MT}} \cup \boldsymbol{D_{SRC}} \cup \boldsymbol{D_{TGT}}$ represent all the available data, where $\boldsymbol{D_{MT}}$ represents the available parallel data and $\boldsymbol{D_{SRC}}$, $\boldsymbol{D_{TGT}}$ represent the source-side and target-side monolingual corpora respectively. Equation 5.1 then takes the following form (under the assumption of the mutual exclusivity of $\boldsymbol{D_{MT}}$, $\boldsymbol{D_{SRC}}$, and $\boldsymbol{D_{TGT}}$):

$$\log p(\boldsymbol{\theta}|\boldsymbol{D}) = \log p(\boldsymbol{D_{MT}}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}|\boldsymbol{D_{SRC}} \cup \boldsymbol{D_{TGT}}) - \log p(\boldsymbol{D_{MT}}) \tag{5.3}$$

Similarly to the original paper $\log p(\boldsymbol{D_{MT}}|\boldsymbol{\theta})$ is the main training objective and $\log p(\boldsymbol{\theta}|\boldsymbol{D_{SRC}} \cup \boldsymbol{D_{TGT}})$ holds the information about the unsupervised LM pretraining. Let $\boldsymbol{\theta_{SRC}} \subset \boldsymbol{\theta}$ and $\boldsymbol{\theta_{TGT}} \subset \boldsymbol{\theta}$ represent the parameters of the encoder and decoder respectively. In the context of unsupervised pretraining, $\boldsymbol{\theta_{tgt}}$ does not contain parameters related to the encoder-decoder attention mechanism. The encoder-decoder attention layers are ignored during the unsupervised pretraining phase, setting their outputs to 0 and using only the residual connections to allow proper information flow. In the subsequent MT fine-tuning phase, the encoder-decoder is initialized randomly, similarly to the standard NMT training practices.

By design, the parameter sets $\boldsymbol{\theta_{SRC}}$ and $\boldsymbol{\theta_{TGT}}$ are mutually exclusive. Also, the source-side and target-side LM pretraining is only conditioned on the source-side and target-side monolingual data respectively. Therefore, the language modeling prior $\log p(\boldsymbol{\theta}|\boldsymbol{D_{SRC}} \cup \boldsymbol{D_{TGT}})$ can be replaced by the following factorization:

$$\log p(\boldsymbol{\theta}|\boldsymbol{D_{SRC}} \cup \boldsymbol{D_{TGT}}) = \log p(\boldsymbol{\theta_{SRC}}|\boldsymbol{D_{SRC}}) + \log p(\boldsymbol{\theta_{TGT}}|\boldsymbol{D_{TGT}}) \quad (5.4)$$

The factorization in Equation 5.4 allows isolated pretraining of subsets of network parameters, possibly using various data source for each. The information learned by the isolated parts of the network, if not forgotten, can be then used for compositional learning in the subsequent fine-tuning phase, allowing few-shot or zero-shot learning.

Similarly to the original EWC paper, the source-side and target-side LM probability distributions cannot be efficiently computed during the NMT fine-tuning phase, requiring a similar Laplace approximation. This results in the following NMT loss:

$$L(\boldsymbol{\theta}) = L_{MT}(\boldsymbol{\theta}) + \frac{\lambda_{SRC}}{2} \sum_{i:\theta_i \in \boldsymbol{\theta_{SRC}}} F_i(\theta_i - \theta_i^*)^2 + \frac{\lambda_{TGT}}{2} \sum_{j:\theta_j \in \boldsymbol{\theta_{TGT}}} F_j(\theta_j - \theta_j^*)^2 \quad (5.5)$$

The parameters $\boldsymbol{\theta}^*$ represent the parameter values at the end of the LM pretraining. Note, that even though we make a distinction between the hyper-parameters λ_{SRC} and λ_{TGT} for individual weighting of the contribution of the pretrained source-side and target-side LM, in the following experiments, we use a single hyper-parameter $\lambda = \lambda_{SRC} = \lambda_{TGT}$.

5.2.2 Experiments: Unsupervised NMT Pretraining

For the low-resource NMT experiments, we used data available for the IWSLT 2018 Basque-to-English machine translation task.[7] The corpora provided by the organizers consist of 5,600 sentence pairs from the TED Talks domain and roughly 940,000 sentence pairs from a general domain. For unsupervised pretraining, we used Basque Wikipedia articles for source-side LM training and NewsCommentary 2015 for the target-side LM.[8] Both monolingual corpora contain around 3M sentences. We used UDPipe Version 1[9] (Straka and Straková, 2017) for sentence splitting and SentencePiece[10] for subword tokenization. The SentencePiece models were trained on the Basque and English monolingual corpora respectively and were later also used to process the parallel data.

[7] `https://sites.google.com/site/iwsltevaluation2018/TED-tasks`

[8] The latter corpus available at `http://www.statmt.org/wmt18/translation-task.html`.

[9] `http://ufal.mff.cuni.cz/udpipe`

[10] `https://github.com/google/sentencepiece`

We used the development data provided by IWSLT 2018 containing 1,140 sentence pairs for evaluation during training. We used the same development data for FIM approximation after the unsupervised LM pretraining. Even though the sentence distribution varies between the development data and the training monolingual corpora, our goal was not to completely preserve the original LM ability of the encoder/decoder – we only required the EWC regularizer to help avoid overfitting on the training data in the subsequent NMT fine-tuning. During the final evaluation, we used the IWSLT 2018 testset provided by shared task organizers, containing 1,051 sentence pairs.

In the following experiments, we investigate, whether the EWC regularization combined with the LM pretraining can be used to mitigate overfitting in the low-resource NMT. We use a Transformer implementation provided by the Neural Monkey framework for neural sequence-to-sequence learning (Helcl and Libovický, 2017).[11] The details about the model hyper-parameters are in Appendix A.4. We separate the source-side and target-side vocabularies and set each one to contain 32k subwords. During the LM pretraining, the output softmax layer of both the encoder and decoder is tied to their respective embedding layers (Press and Wolf, 2017) to reduce the number of trainable parameters. We do the same thing with the decoder during the NMT fine-tuning.

During the LM pretraining phase, we apply identical training hyper-parameters to both the encoder and decoder. We train both using a cross-entropy objective with teacher-forcing, considering only the leftward sequence prefix when generating output tokens. This has a potential drawback when applied to the encoder because it uses both leftward and rightward context during the NMT fine-tuning phase. We discuss the alternative approaches later in this section. As mentioned before, we ignore the encoder-decoder attention layer in the decoder during the pretraining phase and randomly initialize the layer at the beginning of the NMT fine-tuning. Any random parameter initialization is done by sampling values from a standard normal distribution. We reset all training-related parameters (learning rate, momentum, etc.) during the subsequent NMT fine-tuning. We set the EWC regularizer hyper-parameter $\lambda = 0.02$ during the NMT fine-tuning. For efficiency, we estimate the values of FIM using a small holdout dataset; the previous work has shown that it is not necessary to estimate the values using the whole training dataset (Thompson et al., 2019). We pretrain the LM of depth 3 and use it to initialize the first three layers of the respective encoder/decoder NMT architecture - the rest of the model is randomly initialized at the beginning of the fine-tuning.

[11] Our EWC implementation is available at: `https://github.com/ufal/neuralmonkey/tree/ewc_aclsrw2019`.

We compare our EWC regularization approach with the LM-objective regularization (Ramachandran et al., 2017). The latter uses the original LM training objective as an auxiliary objective during the fine-tuning phase. For a more fair comparison, the auxiliary LM objective is only computed using the individual source-side and target-side sentences in the bilingual training data - we do not use the original monolingual corpora during fine-tuning because, in practice, the data from the previously learned tasks are not always available in the later stages of the training. Furthermore, we train a separate MT system without pretraining in each translation direction using the combination of the available in-domain and out-of-domain data and use these models to create additional synthetic data via backtranslation (Sennrich et al., 2016a). Using the original bilingual data and the synthetic data, we train another contrastive NMT model and compare its performance with the EWC-regularized and LM-objective regularized models. We save the top 4 best performing checkpoints for each setup based on their performance on the validation dataset. We compare the performance of the approaches both in the single best-performing and in the ensemble decoding scenario. We ensemble the models by averaging the output log probabilities generated by each model during every decoding step.

Table 5.1 shows the performance comparison between the suggested approaches and EWC-regularized fine-tuning. We also provide the results of a straightforward baseline – a system trained only on the available bilingual data without any regularization. The EWC-regularization slightly outperforms the LM-objective regularizer when applied only on the NMT decoder. Still, both methods prove themselves as inferior to a backtranslation approach which is also easier to deploy in practice.

Additionally, the effect of both regularization methods becomes detrimental when applied to the NMT encoder. This performance drop is likely due to the different nature of the LM pretraining objective (left-to-right decoding) and the fine-tuning (using both left and right context). Figure 5.2 shows the level of performance degradation of the EWC-regularized models with respect to the depth of the pretrained LM. A larger restriction to the NMT encoder (deeper LM) leads to a bigger performance drop possibly due to the incompatibility of the encoder LM pretraining objective and its function during NMT. Using a different pretraining objective, namely the masked language model (MLM, Devlin et al., 2019) could be a more suitable alternative. Lastly, the higher drop in EWC-regularized models, compared to LM-regularization is then likely due to a more restricting nature of the EWC-regularizer, however, this hypothesis requires further investigation.

Due to the EWC-regularizer only slightly outperforming the LM-regularizer in the decoder-only regularization, we also compared the overall training speed of the two approaches. Even though LM-regularization proved to be less detrimental (based on the encoder-only comparison), in theory, an additional model graph execution is

required when computing the LM loss. Just the decoder itself needs to be run in two modes: conditioned on the source side (NMT objective) and conditioned only on its output (LM objective). Figure 5.3 shows that although each model required a roughly similar number of updates to converge during the fine-tuning, the EWC-regularized model required about 2-3 times less wall-clock time to finish training. This comparison was measured on models trained using a single NVIDIA GeForce GTX 1080 Ti GPU.

B

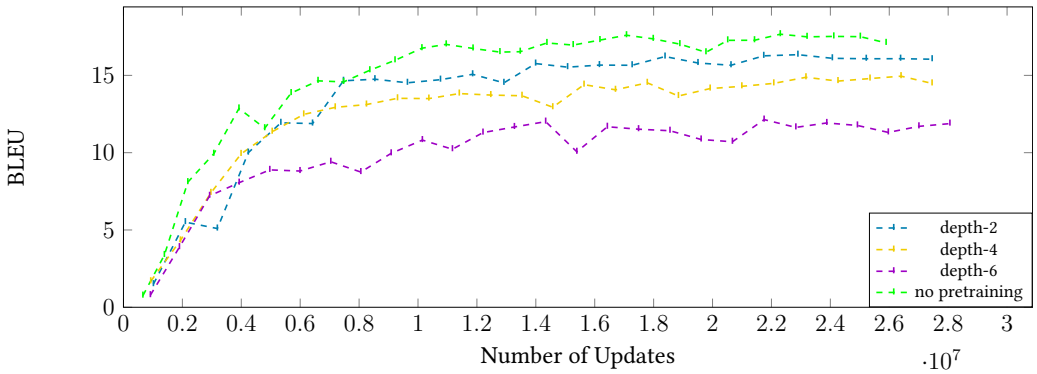

Figure 5.2: NMT model performance with different depth of pretrained encoder LM. The encoder initialized by the LM was later fine-tuned using EWC regularization. The performance of a model trained from scratch is included for comparison. We reproduce the original figure from Variš and Bojar (2019).

		SRC	TGT	ALL
Baseline	15.68	–	–	–
Backtrans.	19.65	–	–	–
LM best	–	13.96	15.56	16.83
EWC best	–	10.77	**15.91**	14.10
LM ens.	–	15.16	16.60	17.14
EWC ens.	–	10.73	**16.63**	14.66

Table 5.1: Comparison of the translation performance of fine-tuned models with the proposed EWC regularization and previous LM regularization. We compare the effects of pretraining encoder-only (*SRC*), decoder-only (*TGT*), and the whole Transformer network (*ALL*). Each pretrained LM contains 3 Transformer layers. We perform a comparison between the single best checkpoint and the model ensemble using the parameter average of the last 4 best checkpoints. Results with the proposed method outperforming previous work are in bold. We reproduce the original table from Variš and Bojar (2019).

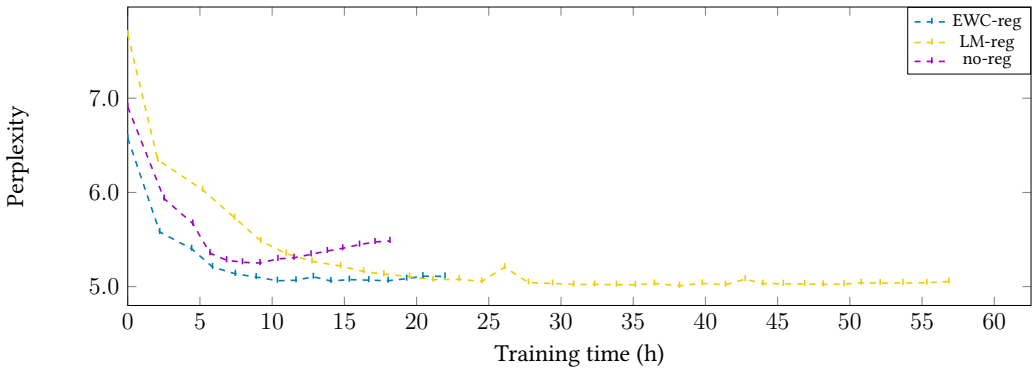

Figure 5.3: Relative model perplexity convergence time comparison of models with pretrained decoder. The compared models used either no regularization (no reg.), LM regularization, or EWC regularization. The models were trained using the same number of training examples, each initialized with a pretrained LM with three Transformer layers. We reproduce the original figure from Variš and Bojar (2019).

5.3 Weight Consolidation Against Catastrophic Forgetting

The results from the EWC-regularization experiments presented in the previous section suggest that, compared to the original EWC prototype experiments covered by Kirkpatrick et al. (2017), the application of EWC to more sophisticated SoTA neural architectures, such as Transformers, is non-trivial. The original supervised learning experiments were investigating the effects of EWC regularization on the optimization of a rather simple, multi-layered perceptron (MLP) in the MNIST experiments and a convolutional network in the reinforcement learning (RL) experiments (based on the architecture proposed by Mnih et al., 2015). Some of the common techniques (e.g. label smoothing, Szegedy et al., 2016) or architectural modifications (e.g. layer normalization, Ba et al., 2016, residual connections, He et al., 2016b) and their influence on the effectiveness of EWC were not investigated in the original publication. Furthermore, their supervised learning experiments focus only on the single datapoint classification (output only conditioned on the input) whereas sequence prediction (output conditioned on input and the previously generated output) arguably poses a bigger challenge.

Still, EWC is not the only viable regularization method aimed at reducing CF. In the context of NMT, the alternatives were investigated mostly in the area of domain adaptation. The related literature (Miceli Barone et al., 2017; Khayrallah et al., 2018) compared various regularization schemes, e.g. using original domain output distribution (Khayrallah et al., 2018), MAP-L2 regularization (Chelba and Acero, 2006) or Bayesian dropout (Gal and Ghahramani, 2016). Later Thompson et al. (2019) demon-

strated that EWC can be used to manage the trade-off between the original general domain translation knowledge and the newly adapted domain-specific knowledge, outperforming the previous approaches. Importantly, these methods were studied in the context of recurrent neural networks, not Transformers.

The supervised learning experiments in the original work on EWC explored only IL scenarios where the original input representations are randomly permuted, creating a dataset for a new task. A more recent survey of the methods aimed at alleviating CF (Kemker et al., 2018) demonstrated that the original positive empirical results of EWC regularization wer related this specific input permutation. In the other IL scenarios, EWC fails at knowledge retention, at least in comparison with the other knowledge retention approaches. To better analyze the effects of the methods for avoiding CF, they suggest following IL scenario taxonomy in the context of classification:[12]

1. **Data Permutation.** The elements of every feature vector are randomly permuted, with the permutation held constant within a session, but varying across sessions. The model is evaluated on its ability to recall data learned in prior study sessions. Each session contains the same number of examples.

2. **Incremental Class Learning.** After learning the base set, each new session learned contains only a single class.

3. **Multi-Modal Learning.** The model incrementally learns different datasets, e.g., learn image classification and then audio classification.

Our main focus in this thesis is textual sequence-to-sequence, thus we will only focus on the Incremental Class Learning problem. Data Permutation experiments are not applicable due to the discrete nature of the input-output, the tokens are represented by the integer-valued indices to the vocabulary and the real-valued embedding matrices are a trainable parameter. We avoid the multi-modal learning experiments to reduce the scope of this thesis.

In the context of sequence learning, we decided to expand the Incremental Class Learning set of tasks into the following sub-categories:

1. **True Incremental Class Learning.** Given a fixed vocabulary, only a subset of vocabulary tokens is present in the training dataset for Task A and a different subset is provided by the Task B training data.

2. **Multi-task Learning.** A similar set of inputs is presented by both Task A and Task B and different task-specific outputs are required for each task. The task choice is indicated on input, e.g. by a special task-indicator token.

[12] We provide the original task definitions proposed by Kemker et al. (2018).

For True Incremental Class Learning, we create the following dataset: given a vocabulary of tokens $V = \{A, B, C, D\}$, we create two separate datasets containing only sequences of A, B tokens (`A-B`) and C, D tokens (`C-D`), respectively. Both datasets contain 30k unique input sequences of up to 20 tokens. The datasets are then randomly split into a train, validation, and test dataset with 28k, 1k, and 1k input sequences respectively. We set up two types of experiments, one with the `copy` operation (copy the input sequence to the output) and one with the `reverse` operation (generate the input sequence in the reverse order). Based on the given task, we algorithmically generate the correct output sequence for each dataset split. Since there is only a single task involved during the True Incremental Class Learning, we do not add the task-indicating token at the beginning of the input. The goal of this IL task is to first learn the algorithm on the `A-B` sequences and then fine-tune the model on the `C-D` sequence, measuring the performance on both respective testsets.

One-to-many multilingual NMT can be considered an example of Multi-task Learning. Having a single input sentence and a target-language indicator token, we can consider translation into different languages being different tasks, expecting task-specific output (translation in a given language) given an identical input sentence.

The original survey results demonstrate that EWC performs quite poorly in the Incremental Class Learning classification (Kemker et al., 2018). In the following sections, we provide a study of EWC regularization with respect to various the two subtasks of Incremental Class Learning in sequence-to-sequence Transformers. We study the problems on the string editing task and the task of multilingual NMT.

5.3.1 FIM Normalization and EWC Stabilization

A crucial component of the EWC regularization is computing the diagonal of the FIM. In practice, due to not having access to the data generating distribution $p_A(\boldsymbol{y}|\boldsymbol{x})$ for Task A, the computation of FIM is intractable. Instead, we estimate FIM using empirical Fisher (Schraudolph, 2002b) by computing a sample mean of the square of the gradients using a dataset sampled from the said distribution:

$$F(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim p_A}[\nabla \log p_\theta(\boldsymbol{y}|\boldsymbol{x}) \nabla \log p_\theta(\boldsymbol{y}|\boldsymbol{x})^\intercal] \tag{5.6}$$

$$F(\boldsymbol{\theta}) \approx \frac{1}{N} \sum_{i}^{N} \nabla \log p_\theta(\boldsymbol{y_i}|\boldsymbol{x_i}) \nabla \log p_\theta(\boldsymbol{y_i}|\boldsymbol{x_i})^\intercal \tag{5.7}$$

The resulting FIM estimate can vary between tasks which can make hyper-parameter tuning of the EWC regularization hyper-parameter λ an irritating chore. Furthermore, the FIM values can also vary based on the chosen notion of sequence loss: we can either optimize the model to increase the log-likelihood of the training sequences as a whole (represented as a sum of log-likelihoods of the sequence tokens) or we can take individual predicted tokens as separate training targets. The original normalization of the summed squares of gradients in FI ($\frac{1}{N}$) accounts for the varying sizes of the available sample, however, it does not consider the differences between the various tasks. In contrast, to further simplify the λ hyper-parameter search, we suggest normalization of the estimated FIM. Note that even though the normalization can change the properties of the measured FI, the normalization should not affect the EWC-regularization term. Given a normalization value Z, we get the following equation:

$$\frac{\lambda}{2}\sum_i F_{i,A}(\theta_i - \theta^*_{i,A})^2 = \frac{\lambda Z}{2}\sum_i \frac{F_{i,A}}{Z}(\theta_i - \theta^*_{i,A})^2 \tag{5.8}$$

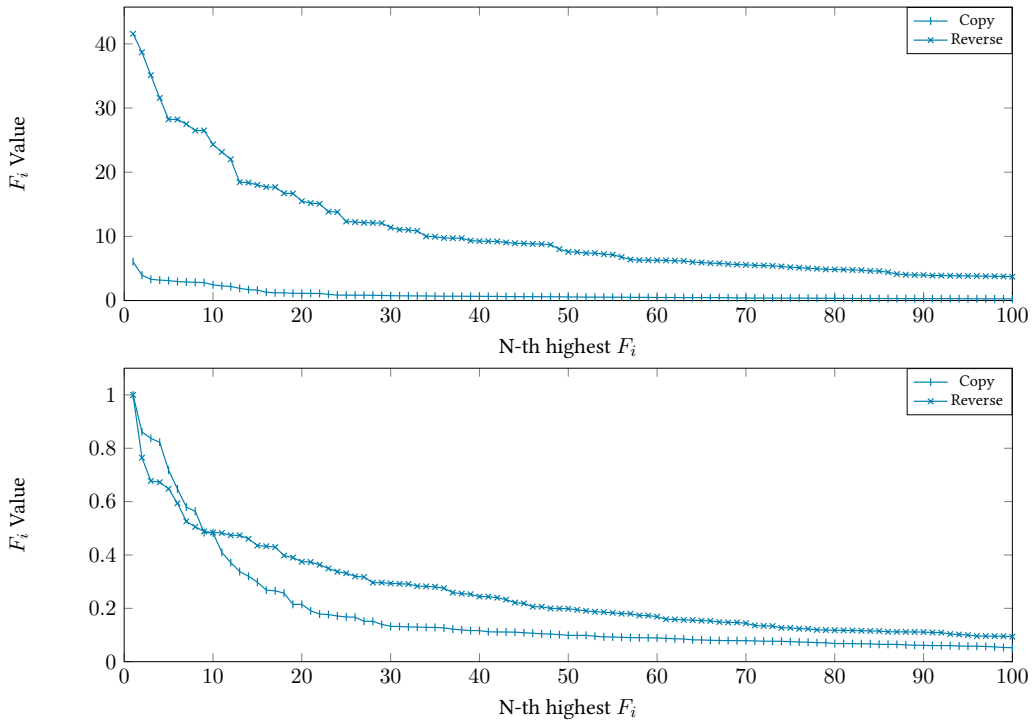

Figure 5.4: Top 100 values of the diagonal of the empirical FIM in the context of different tasks (`copy`, `reverse`) using only the `A-B` sequences during training. **Top**: Normalization using the sample size. **Bottom**: Normalization by the highest FIM diagonal value.

For our experiments, we propose normalization value Z to be the highest $F_{i,A}$ value of a given model trained for Task A, setting the FI values to the interval $(0, 1)$. Figure 5.4 illustrates the differences between the original $\frac{1}{N}$ normalization and the proposed *max* normalization showing the top-100 values of the diagonal of FIM.[13] Using the sample size normalization can lead to a significantly different scaling of FI depending on a given task (Figure 5.4, top). On the other hand, the max-value normalization results in the FI values being in a similar interval while still preserving the relative differences between the original values (Figure 5.4, bottom).

There is another issue regarding FI: extremely high values of FI can potentially hurt knowledge retention. Going back to Equation 5.2, the update size of the network parameter θ_i has the following form:

$$\Delta\theta_i = -\alpha\frac{\partial L}{\partial\theta_i} - \alpha\lambda F_{i,A}(\theta_i - \theta^*_{i,A}) \tag{5.9}$$

As pointed out by Kutalev and Lapina (2021), the size of the parameter update with respect to the regularization term $-\alpha\lambda F_{i,A}(\theta_i - \theta^*_{i,A})$ is scaled with the size of the hyper-parameter λ and FI $F_{i,A}$. Furthermore, $-\alpha\lambda F_{i,A} > 1$ results in *over-shooting* the optimal Task A parameter $\theta^*_{i,A}$, and if $-\alpha\lambda F_{i,A} > 2$, the distance of the parameter θ_i from the optimal value for Task A even increases, basically *forgetting* the task. Given the huge differences between the sample-normalized FI of the Transformer parameters (Figure 5.4, top), a wrong choice of the hyper-parameter λ can result in either ignoring the effect of parameters with too low FI value or even potentially accelerating the *forgetting* process by forcing important parameters to move away from their Task A optimum.

Instead of normalizing the values of the FIM diagonal, Kutalev and Lapina (2021) suggest modifying the original EWC regularization term to normalize the values of the gradient. Inspired by their proposal, we suggest the following Fisher-normalized, EWC regularizer:

$$L(\theta) = L_B(\theta) + \frac{\lambda}{2}\sum_i \frac{F_{i,A}}{F_{i,A}+1}(\theta_i - \theta^*_{i,A})^2 \tag{5.10}$$

The EWC regularization term from Equation 5.10 has the following gradient:

$$-\alpha\lambda\frac{F_{i,A}}{F_{i,A}+1}(\theta_i - \theta^*_{i,A}) \tag{5.11}$$

[13] On both copy and reverse tasks, the network was able to reach perfect accuracy on the initial task. We used the validation datasets to estimate the FIM for the respective tasks.

As a result, the sole contribution of the regularizer gradient to the parameter update cannot exceed the distance between the current parameter value and the value optimized for the previous task due to high values of $F_{i,A}$. Compared to Kutalev and Lapina (2021), we only normalize the FI values to stabilize the training. Hyper-parameters α and λ are not derived from the data and can be adjusted as required. In the following sections, we conduct experiments with the FIM normalization and EWC stabilization and compare them to the original approach.

5.3.2 Experiments: String Editing

Task	Input	Output
Copy (`A-B`)	a b a b	a b a b
Copy (`C-D`)	c d d c c	c d d c c
Reverse (`A-B`)	a a b a b	b a b a a

Figure 5.5: Input and example for `copy` and `reverse` using different vocabulary subsets for True Class Incremental learning (`A-B`, `C-D`). The former subset is used for pretraining, the latter for model fine-tuning.

First, we evaluate the normalization methods on the True Incremental Class Learning string editing task. We take the datasets with shared vocabulary $V = \{A, B, C, D\}$ for `copy` and `reverse` task described earlier. For each task, we first train a model on a subset of data containing only tokens A and B and then fine-tune it on a subset with tokens C and D. We compare the original *sample* normalization with the proposed *max* normalization, i.e. FIM values normalized to the $(0, 1)$ interval. We combine each normalization method with the *original* EWC regularizer and the *stabilized* variant described in Equation 5.10. Figure 5.5 show examples from the training dataset. Similarly to Section 4.1.1, during the evaluation, we measure string-level accuracy $ACC = \frac{n_{correct}}{n_{all}}$, where $n_{correct}$ is the number of exact matches of the produced string and the reference output.

The details about the Transfomer hyper-parameters are in Appendix A.5. All fine-tuning attempts we initialized by the same pretrained `A-B` model and only differ in the selected EWC method and hyper-parameter λ. We fine-tuned each model for the same number of updates and picked the model checkpoint that had the best performance on the combined `A-B` and `C-D` validation data. In our preliminary experiments, we found that λ values close to 1000 work reasonably well on the copy task as demonstrated in Figure 5.6 (top). We measured the effects of various values of the hyper-parameter $\lambda \in (500, 1500)$ in combination with the normalization and stabi-

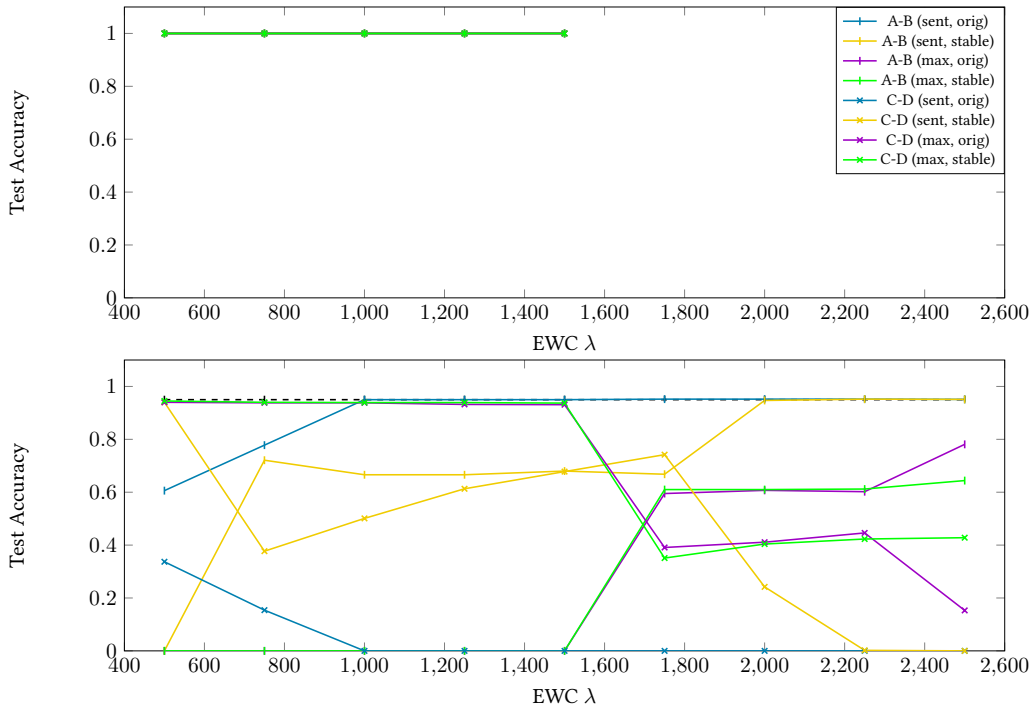

Figure 5.6: Accuracy comparison between the various combinations of FIM estimation (*sent*, *max*) and the EWC regularization terms (*orig*, *stable*). The models fine-tuned with different values of the hyper-parameter λ were evaluated using our incremental class learning benchmark – the original task (`A-B`) was used for model pretraining and the follow-up task (`C-D`) was used during fine-tuning. **Top**: String copying (`copy`) task; all tasks reaching perfect accuracy. **Bottom**: String reversal (`reverse`) task. Models were initially trained on examples with a, b tokens and later fine-tuned on the examples with c, d tokens. The black dashed line indicates the test accuracy of the model used for initialization of the `reverse` model fine-tunning, measured on the initial `A-B` task validation data.

lization methods for the copy task. Later we run the same experiments on the reverse dataset using values $\lambda \in (500, 2500)$,[14] to see whether the results are consistent and whether the normalization enables hyper-parameter selection from a similar interval of values.

Figure 5.6 (bottom) shows the EWC performance on the reverse task concerning various regularizer hyper-parameter values. Compared to copy, incremental learning becomes a trade-off between the knowledge of the original task (`A-B`) and the subsequent task (`C-D`). The results show that the original approach truly requires a much wider exploration of the hyper-parameters values. The methods using *max* normalization of FIM offer a better trade-off than the original approach (sent, orig) on the explored hyper-parameter value interval, however, it still leads to sacrificing

[14]We expanded the investigated interval due to some of the methods performing poorly on values from the original interval.

performance on one task for the sake of the other. The combination of the proposed normalization and stabilization, however, seems redundant. Interestingly, the stabilization of the FI gradient in the EWC regularizer (sent, stable) leads to much better results ($\lambda \in (1200, 1800)$), keeping both accuracies above 50%.

Based on these results, we conclude, that the regularizer stabilization can improve the effects of EWC and it does not require FI normalization. In the next section, we will measure the effects of the normalization and stabilization methods in the multi-task setting.

5.3.3 Experiments: Multilingual NMT

To see the effects of the proposed EWC modifications on incremental multi-task learning (MTL), we compare them in the context of the multilingual incremental NMT. Similarly to our unsupervised pretraining experiments, we choose a scenario of adapting models trained on high-resource NMT and fine-tuning them for low-resource languages. A potential benefit of avoiding CF can be storage reduction – in comparison to the standard fine-tuning approaches, due to a model not forgetting the original high-resource translation, we would not require to store the original pretrained model separately.

In the following experiments, we use the OPUS-100[15] dataset for multilingual machine translation (Zhang et al., 2020), sampled from the OPUS collection (Tiedemann, 2012). The OPUS-100 dataset is English-centric, meaning that any of the training language pairs contain English on either source or target side. The dataset contains 100 languages of varying corpus sizes, the largest ones are limited to 1M training examples. Each language pair dataset was randomly sampled from the original OPUS collection without any curation besides the cross-lingual deduplication[16] which is sometimes reflected in the quality of the training samples. Based on the dataset sizes, the sampled corpora are separated into high-resource (up to 1M training examples) and low-resource (up to 100k or 10k training examples). Furthermore, the authors put aside a sample of 2k validation and test examples for each language pair and create a set of zero-shot evaluation corpora for direct translation between selected languages excluding English.

Inspired by Zhang et al. (2020), we choose four typologically distinct languages with varying training dataset sizes: German (De), traditional Chinese (Zh), Breton (Br), and Telugu (Te). Table 5.2 summarizes the basic statistics about the training corpora used in our experiments. In our experiments, we consider the De, Zh corpora as high-resource and Br, Te as low-resource.

[15] https://opus.nlpl.eu/opus-100.php

[16] Meaning there are no duplicates of English-side sentences between the different language pairs.

	Num. of Sentences	Num. of Tokens $\{De, Zh, Br, Te\}$	Num. of Tokens En
De-En	1,000,000	10,860,823	11,636,324
Zh-En	1,000,000	1,703,534	16,702,315
Br-En	153,447	684,248	591,374
Te-En	64,352	263,810	312,283
Total	2,217,799	13,512,415	29,242,296

Table 5.2: Summary of the training dataset sizes used in the multilingual experiments.

We compare our multilingual fine-tuning approach using EWC regularization with the standard fine-tuning that avoids using any form of regularization. Furthermore, we compare our results to the bilingual baselines, trained using each language pair separately, and a joint multilingual baseline trained on the full combination of the available training corpora. We compare the performance of the listed models on both many-to-one (to English) and one-to-many (from English) translations. We avoid any language-specific tokenization of the corpora due to the differences between the individual language scripts. Instead, we apply byte-pair encoding (BPE) on the concatenation of all available training corpora and create a single subword segmentation scheme with 64k merges. In the following experiments, we always combine the source- and target-side vocabularies. Inputs to the models translating into multiple languages are prefixed with an additional target-language token indicating to which language the input needs to be translated.

The details about the default model hyper-parameters are in Appendix A.6. We train the bilingual models and low-resource models (*En-Low*, *Low-En*; containing both Br and Te) for 200k updates. We train the high-resource models (*En-High*, *High-En*; containing De and Zh) for 600k updates. Similarly, we train the multilingual models (*En-All* and *All-En*; containing all four languages) for 600k updates. All fine-tuning experiments are initialized with their respective high-resource model and tuned on the low-resource datasets for additional 200k updates. With respect to EWC, we compare the original FIM sample normalization (*sent*) with the proposed max-value normalization (*max*) in combination with the original EWC regularization term (*orig*) and the "stabilized" version described in Equation 5.10 (*stable*). Due to the computational cost of the training (several days per model in Fairseq), we choose only a single EWC hyper-parameter $\lambda = 1800$ for all regularized model variants based on the performance in the string editing experiments. In the final evaluation, the best-performing parameter checkpoint (measured on the combined validation dataset) is used to compare the individual model variants.

We evaluate the models using BLEU (Papineni et al., 2002). Due to BLEU having well-known flaws (Callison-Burch et al., 2006; Lin and Och, 2004), we also use the recently proposed COMET metric (Rei et al., 2020), a Transformer-based scoring model for NMT evaluation. Even though the correctness of the COMET metric is difficult to analyze (it is an optimized deep learning model, with possible data-derived biases), it has been reported to have a much better correlation with human judgement than previous n-gram precision metrics, such as BLEU. We use the standard COMET version requiring reference translations (*Cmt*) and a referenceless evaluation model (Cmt_{QE}).

ID	System	En-De			En-Zh			En-Br			En-Te		
		BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}
1	base En-De	29.5	0.1600	0.1090	–	–	–	–	–	–	–	–	–
2	base En-Zh	–	–	–	26.1	0.2342	0.1113	–	–	–	–	–	–
3	base En-Br	–	–	–	–	–	–	17.3	-0.4024	0.0907	–	–	–
4	base En-Te	–	–	–	–	–	–	–	–	–	20.3	0.0963	0.0973
5	base En-All	28.4	0.1189	0.1081	29.5	0.2806	0.1116	17.5	-0.3121	0.0913	22.0	0.2723	0.0983
6	base En-Low	–	–	–	–	–	–	15.9	-0.4209	0.0907	18.4	0.0348	0.0991
7	base En-High	28.3	-1.3560	0.1079	29.6	0.2840	0.1118	–	–	–	–	–	–
8	(7) ⇒ En→Low	1.4	-1.4218	0.0851	0.5	-1.5117	0.0817	**21.8**	**-0.2186**	0.0905	**29.0**	**0.3447**	0.0991
9	(7) ⇒ En→Br	1.2	-1.5668	0.0819	0.3	-1.6798	0.0784	**22.5**	**-0.2058**	0.0907	–	–	–
10	(7) ⇒ En→Te	0.7	-1.4740	0.0884	0.3	-1.5388	0.0850	–	–	–	**30.9**	**0.3593**	0.0987
11	(7) ⇒ En→Low (sent, original)	19.9	-0.5722	0.0891	1.1	-0.7708	0.0838	6.6	-1.0831	0.0915	12.5	-0.0934	0.0958
12	(7) ⇒ En→Low (sent, stable)	16.9	-0.9108	0.0789	1.0	-1.1127	0.0740	9.0	-0.6762	0.0881	14.0	-0.0341	0.0967
13	(7) ⇒ En→Low (max, orig)	16.1	-0.8169	0.0849	0.3	-1.1253	0.0770	14.7	-0.3902	0.0906	19.8	0.1678	0.0980
14	(7) ⇒ En→Low (max, stable)	10.7	-1.1039	0.0827	4.0	-0.9737	0.0844	17.9	-0.2839	0.0905	22.0	0.2605	0.0985
15	(7) ⇒ En→Br (sent, orig)	19.6	-0.7011	0.0827	0.2	-1.3054	0.0658	7.7	-0.7525	0.0901	–	–	–
16	(7) ⇒ En→Br (sent, stable)	18.7	-0.7686	0.0822	0.6	-1.0649	0.0707	8.1	-0.6978	0.0906	–	–	–
17	(7) ⇒ En→Br (max, orig)	15.1	-0.8617	0.0869	1.7	-0.8167	0.0875	15.1	-0.3737	0.0909	–	–	–
18	(7) ⇒ En→Br (max, stable)	13.8	-0.9255	0.0861	1.3	-0.9365	0.0846	15.3	-0.3724	0.0903	–	–	–
19	(7) ⇒ En→Te (sent, orig)	19.2	-0.5353	0.0930	4.4	-1.0517	0.0831	–	–	–	15.9	0.0455	0.0972
20	(7) ⇒ En→Te (sent, stable)	17.5	-0.6401	0.0897	4.1	-1.1839	0.0817	–	–	–	17.0	0.1160	0.0979
21	(7) ⇒ En→Te (max, orig)	11.2	-1.0004	0.0815	2.1	-1.3131	0.0845	–	–	–	**26.2**	**0.3692**	0.0990
22	(7) ⇒ En→Te (max, stable)	9.7	-1.0740	0.0804	2.0	-1.3422	0.0838	–	–	–	**26.5**	**0.3356**	0.0990

Table 5.3: Comparison of one-to-many translation models. We compare bilingual (1-4) and jointly optimized (5-7) baselines with models fine-tuned without any regularization (8-10) and models fine-tuned using EWC with different normalization approaches (11-22). We compare the sample-level (*sent*) and max-value (*max*) normalization of FIM in combination with the original (*orig*) and the stabilized (*stable*) variant of the EWC regularizer. Each model fine-tuning was initialized by the high-resource multi-lingual model (7). Fine-tuned models that outperform the jointly trained multilingual baseline (En→All) on a given language are in **bold**.

Table 5.3 shows a comparison in the context of the one-to-many translation task. This translation direction can be considered multi-task learning, with each target-side language representing an individual task (identified by the special input token). The regularized models did not perform as well on the low-resource languages as the models using fine-tuning without any regularization. However, they suffered much less from the catastrophic forgetting which completely removed the ability of the unregularized fine-tuned models to translate into the high-resource languages.

ID	SysTem	De-En			Zh-En			Br-En			Te-En		
		BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}
1	base De→En	32.5	0.2374	0.1107	–	–	–	–	–	–	–	–	–
2	base Zh→En	–	–	–	38.9	0.3234	0.1143	–	–	–	–	–	–
3	base Br→En	–	–	–	–	–	–	15.9	-0.3841	0.0973	–	–	–
4	base Te→En	–	–	–	–	–	–	–	–	–	24.8	0.0372	0.1029
5	base All→En	31.1	0.2266	0.1117	39.1	0.3370	0.1149	18.3	-0.2328	0.0995	30.2	0.2121	0.1059
6	base Low→En	–	–	–	–	–	–	16.0	-0.3816	0.0968	25.3	0.0152	0.1040
7	base High→En	32.0	0.2363	0.1113	39.0	0.3453	0.1148	2.3	-1.3128	0.0885	1.3	-0.9843	0.1139
8	(7) ⇒ Low→En	1.1	-1.4106	0.0930	0.0	-1.7100	0.0873	**21.0**	**-0.1809**	0.0991	**35.9**	**0.2671**	0.1067
9	(7) ⇒ Br→En	0.3	-1.4640	0.0932	0.0	-1.7131	0.0937	**21.4**	**-0.1829**	0.0988	–	–	–
10	(7) ⇒ Te→En	1.0	-1.5057	0.0886	0.2	-1.7350	0.0871	–	–	–	**36.7**	**0.3011**	0.1061
11	(7) $\overset{EWC}{\Longrightarrow}$ Low→En (sent, orig)	30.4	0.1943	0.1109	38.3	0.3237	0.1145	9.6	-0.6320	0.0968	22.6	0.0511	0.1067
12	(7) $\overset{EWC}{\Longrightarrow}$ Low→En (sent, stable)	30.5	0.1847	0.1112	37.6	0.2962	0.1140	9.6	-0.6276	0.0964	22.6	0.0467	0.1060
13	(7) $\overset{EWC}{\Longrightarrow}$ Low→En (max, orig)	22.7	-0.1198	0.1072	26.8	-0.0381	0.1089	12.4	-0.4561	0.0976	24.8	0.1017	0.1048
14	(7) $\overset{EWC}{\Longrightarrow}$ Low→En (max, stable)	23.0	-0.1329	0.1066	26.7	-0.0458	0.1087	12.8	-0.4671	0.0976	25.7	0.1179	0.1053
15	(7) $\overset{EWC}{\Longrightarrow}$ Br→En (sent, orig)	29.9	0.1688	0.1110	37.7	0.3009	0.1145	9.9	-0.6033	0.0975	–	–	–
16	(7) $\overset{EWC}{\Longrightarrow}$ Br→En (sent, stable)	29.5	0.1510	0.1107	37.4	0.2689	0.1135	10.1	-0.5977	0.0979	–	–	–
17	(7) $\overset{EWC}{\Longrightarrow}$ Br→En (max, orig)	22.4	-0.1514	0.1063	26.6	-0.0557	0.1087	13.8	-0.3960	0.0984	–	–	–
18	(7) $\overset{EWC}{\Longrightarrow}$ Br→En (max, stable)	21.7	-0.1902	0.1053	26.1	-0.0979	0.1068	13.8	-0.4283	0.0982	–	–	–
19	(7) $\overset{EWC}{\Longrightarrow}$ Te→En (sent, orig)	31.3	0.2020	0.1104	38.2	0.3092	0.1140	–	–	–	24.0	0.0972	0.1051
20	(7) $\overset{EWC}{\Longrightarrow}$ Te→En (sent, stable)	31.2	0.2055	0.1105	38.2	0.3183	0.1143	–	–	–	24.0	0.0848	0.1051
21	(7) $\overset{EWC}{\Longrightarrow}$ Te→En (max, orig)	23.2	-0.1205	0.1058	26.4	-0.0322	0.1088	–	–	–	27.6	0.1593	0.1060
22	(7) $\overset{EWC}{\Longrightarrow}$ Te→En (max, stable)	23.2	-0.1222	0.1055	26.5	-0.0418	0.1087	–	–	–	27.1	0.1835	0.1058

Table 5.4: Comparison of many-to-one translation models. We compare bilingual (1-4) and jointly optimized (5-7) baselines with models fine-tuned without any regularization (8-10) and models fine-tuned using EWC with different normalization approaches (11-22). We compare the sample-level (*sent*) and max-value (*max*) normalization of FIM in combination with the original (*orig*) and the stabilized (*stable*) variant of the EWC regularizer. Each model fine-tuning was initialized by the high-resource multi-lingual model (7). Fine-tuned models that outperformed the jointly trained multilingual (All→En) baseline are in **bold**.

This knowledge retention was manifested only on German – translation into Chinese was forgotten, possibly due to not seeing any tokens from the Chinese script during fine-tuning. The models using max-value normalization of FIM ((13), (17), (21)) lead to much better performance on the low-resource NMT than the sample-normalized model; their performance is only slightly dropping in the high-resource translation. However, we suspect that this could be a result of the lack of costly hyper-parameter tuning. Interestingly, the max-value models were still able to perform on par with the bilingual baseline while manifesting at least a small retention on translation to German.

Table 5.4 compares the results of the many-to-one translation. This task is similar to the true incremental class learning from the previous section In this case, however, there is an overlap between the task vocabularies. Again, the results show that the EWC-based methods cannot outperform the unregularized baselines on the low-resource languages. However, they are much more effective at avoiding CF of the original high-resource language translation, including translation from Chinese.

This suggests that, combined with the one-to-many translation results, the exposure to the output vocabulary of the original task plays an important role in remembering the task. The trade-off between the high- and low-resource language performance suggests, that the method using the original sample FIM normalization is more effective in this setting, although, this could again be a consequence of not performing a thorough hyper-parameter search.

Overall the experiments presented in this chapter suggest that given some trade-offs, the EWC regularization method can perform reasonably well on the sequential task incremental learning. Our experiments support results from the previous work on domain adaptation by Thompson et al. (2019) that showed a similar performance trade-off between the original and the new translation task (domain).

5.4 Conclusions

In this chapter, we experimented with the regularization method based on EWC and its effect on various IL tasks. Results of the experiments helped us answer **Research Question 2**: *Can the selective parameter regularization lead to improvements in Transformer performance in incremental learning?* We subdivided the question into three sub-questions and the provide following answers based on our empirical findings.

RQ2.1 *Can selective parameter regularization improve the NMT performance when fine-tuning models initialized by pretrained language models?*

We demonstrated in Section 5.2 that the original idea behind EWC can be broken down to motivate incremental learning in parts of the network separately. Based on this factorization, we proposed using EWC for fine-tuning models initialized by pretrained language models, using the regularization to avoid overfitting on low-resource NMT tasks. The results suggest that this approach can be a decent alternative to the previous LM regularization objective, resulting in comparable model performance while reducing the computation time of the fine-tuning process. However, our method did not surpass the more popular backtranslation approach.

RQ2.2 *How effective is selective parameter regularization for the different classes of incremental learning tasks?*

We proposed a subcategorization of the Incremental Class Learning suggested in previous work (Kemker et al., 2018) to better analyze incremental learning methods in the context of sequence-to-sequence learning. We demonstrated on the True Incremental Class Learning that EWC-regularized Transformer can perfectly learn

both tasks only on a very trivial instance (copy). In harder instances (reverse), the use of EWC and the choice of the regularization hyper-parameter leads to a trade-off between the original and the new task. We later confirmed our findings on the multilingual NMT presenting similar results.

RQ2.3 *Does selective parameter regularization improve model performance in incremental multilingual translation?*

The results presented in Section 5.3.3 demonstrated that the EWC regularizer, when used to adapt high-resource NMT models to low-resource languages, provides a trade-off between the ability to translate in these two domains. This comes at a cost of lower performance on the low-resource languages, compared to the unregularized fine-tuning. On the other hand, the EWC-regularized fine-tuning was still able to reach the performance of the bilingual models trained in the low-resource languages.

6

Transformer Modularization

The ability to learn new concepts and find novel connections within the acquired knowledge is another key aspect of human-like learning (Murphy, 1988; Osherson and Smith, 1981). In cognitive linguistics, a popular hypothesis suggests that by learning only a finite set of production rules, humans have the ability to produce an infinite number of grammatically correct sentences (Chomsky, 1965). If we consider each production rule a snippet of knowledge (which is an oversimplification), the hypothesis implies that the human mind can combine these snippets into much larger building blocks, being able to create possibly an infinite number of novel sentences.

A different view can be offered by the studies of the human brain. Previous analysis of neural activation in the human brain has demonstrated that only a few parts of the brain are active at a given time when faced with different input stimuli (Ramezani et al., 2014). This implies that the brain neurons can specialize for specific tasks/activities and a certain level of knowledge composition happens by activating various subsets of these neurons. Although the role of each individual neuron in this specialized storage of knowledge (in a broad sense) is uncertain, some distinctions of parts human brain and assumptions about their roles can be made (Sporns and Betzel, 2016).

Such behavior is foreign in the current standard Transformer networks – regardless of the network input all parts of the network are active at the same time, contributing to the network output. Although the network architecture does contain modular blocks, the multi-head attention (MHA), the contribution of individual attention heads to the block output is fixed and not conditioned on the current input, i.e. the output of the MHA block is always a combination of all attention heads. Still, the lack of explicit modularity (conditioned by the current input) is not crucial for learning multiple tasks and various aspects of data as demonstrated by the previous

research on Transformers in the area of multi-task learning (MTL), e.g. multilingual LM (Conneau and Lample, 2019b; Chi et al., 2022) or multilingual NMT (Liu et al., 2020; Escolano et al., 2021). The ability to capture these different aspects of data can most likely be attributed to the large feature spaces that these networks operate on and the transformations across these feature spaces.

A certain level of modularity, i.e. non-overlapping distribution of knowledge about different tasks, can be enforced for example by applying task-specific masks (Csordás et al., 2021). While task-specific masking can help to avoid the interference of knowledge about different tasks, it does not enable knowledge composition (Csordás et al., 2021). We hypothesize that one of the reasons behind this inability of the neural networks to combine previous tasks' knowledge is due to a lack of an explicit control mechanism that would allow the network to correctly combine partial knowledge about previously learned tasks.

Such computation control can be provided by conditional neural network computing (Bengio, 2017; Bengio et al., 2013; Rosenbaum et al., 2018). One of the key ideas, training a set of separate, specialized network modules and combining their outputs, is adjacent to mixture-of-experts (MoE, Shazeer et al., 2017; Eigen et al., 2014; Jacobs et al., 1991). The input-conditioned control mechanism (e.g. gating network) provides a set of weights, that enables varying the *mixture* of the experts' outputs through their weighted sum, leading to a specialization of these experts. The shortcoming of MoE is that the whole network is still required during computation since the control mechanism does not usually allow zero weights to keep the control function differentiable and, consequently, easy to train via backpropagation. Later approaches relax even this restriction by introducing noisy top-k gating (Bengio et al., 2013), resulting in a controllable sparsity of the mixture (Shazeer et al., 2017).

As an alternative to the "soft" mixture approach, it is possible to select desired submodules by explicitly masking them. The module mask prediction can be, for example, done by learning an activation-dependent policy using reinforcement learning that decides which expert blocks to use for the computation (Bengio et al., 2016). However, these methods are prone to module collapse, lacking diversity when choosing experts during training often due to self-reinforcing of the favored modules by training them more rapidly (Kirsch et al., 2018). To counter this, some form of regularization needs to be enforced during the training.

Addressing these issues, Kirsch et al. (2018) suggest using a stochastic selection of a subset of modules instead of mixtures. They treat the module subset choice as a latent variable requiring summation over all possible subsets to generate output distribution. To avoid computational explosion during training, they use a generalized Expectation-Maximisation (EM, Neal and Hinton, 1998). In the estimation step, they sample a small number of module subsets based on the current *module proba-*

bility distribution and then choose the best candidate subset by maximizing the *output probability* of the current training label. In the maximization step, they use the best candidate module subset to compute the loss on several data points and update both *module probability* and *output probability* distribution by adjusting the network weights.

Still, module masking requires a pre-designed network structure with maskable modules, leaving us with inductive bias introduced by the architecture design. Instead, Rosenbaum et al. (2018) suggest having a separate set of network modules and a routing network that chooses a subset of these and the order in which they process given input, reducing this inductive bias created by network design.

Lastly, although dropout (Srivastava et al., 2014) can be considered a modularization technique (in the sense of switching of sets of neurons), we do not consider it control because a) it is used mainly during training, b) it is random, thus, not conditioned by the network input.

6.1 Modular Transformer

We propose a modification to the current Transformer network that introduces conditional stochastic computation. We introduce stochasticity into the existing Transformer architecture through stochastic modular layers. Generally, a *modular layer* consists of a set of M modules (subnetworks) representing learnable functions $\boldsymbol{f} = \{f_1, .., f_M\}$, each operating over the layer input x. Given a set of binary masks $\boldsymbol{\xi} = \{\xi_1, .., \xi_M\}$, the output of the modular layer is a masked sum of the module outputs:[1]

$$y = \sum_{i=1}^{M} \xi_i f_i(x) \tag{6.1}$$

Now let us consider a latent variable $\boldsymbol{a} \in 2^{\{1,..,M\}}$ indicating a selection of layer modules such that $\xi_i = \mathbf{1}_{\boldsymbol{a}}(i)$, where $\mathbf{1}_{\boldsymbol{a}}$ is the indicator function. A *stochastic modular layer* is an extension of the modular layer with the *controller* network which estimates a probability distribution $p(\boldsymbol{a}|x)$ of the latent variable $\boldsymbol{a}$, given the layer input x. The network output thus becomes conditioned on both the input sequence x and the module selection $\boldsymbol{a}$. Let $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ be the parameters of the modular network and the controller network respectively. The modular network estimates the probability of the random variable y, given x in the following way:

$$p(y|x, \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{\boldsymbol{a}} p(y, a|x, \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{\boldsymbol{a}} p(y|x, a, \boldsymbol{\theta}) p(a|x, \boldsymbol{\phi}) \tag{6.2}$$

[1] Related work also suggests concatenation (Kirsch et al., 2018).

The resulting training objective is to maximize the log-likelihood of the training data:

$$L(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{x,y \in \boldsymbol{D}} \log p(y|x, \boldsymbol{\theta}, \boldsymbol{\phi}) \tag{6.3}$$

However, as pointed out by Kirsch et al. (2018), with the increasing number of network modules, the summation in Equation 6.2 becomes intractable.[2] Instead of maximizing the log-likelihood directly, they maximize the evidence lower bound (ELBO; also known as the variational lower bound; Kingma and Welling, 2014) on the likelihood using a separate variational distribution $q(\boldsymbol{a})$ that can be independent of the layer input:

$$L_{ELBO} = \mathbb{E}_{q(\boldsymbol{a})}[\log p(y|x, \boldsymbol{\theta}, \boldsymbol{\phi})] + \mathbb{H}[q] \tag{6.4}$$

The resulting gradient of the expectation in Equation 6.4 still requires summation over all module subsets. For this reason, the authors suggest using a Viterbi expectation-maximization (EM) algorithm (Neal and Hinton, 1998) to maximize the lower bound through approximation. They argue that their proposed approach is more efficient at avoiding *module collapse* (Rosenbaum et al., 2018; Shazeer et al., 2017; Bengio et al., 2016), which was the main problem with previous reinforcement learning (RL) approaches, such as REINFORCE (Williams, 1992). However, the efficiency of the approach depends on the size of the E- and M-steps, where increasing the sizes leads to slower training convergence. In our initial experiments, we found the fine-tuning of the EM hyper-parameters to be the most cumbersome part of training.

On the other hand, the sampling approach and the EM optimization are the consequence of the assumption that the module controller is a discrete non-differentiable function. In the next section, we describe our proposed controller inspired by discrete sampling based on Gumbel-softmax (Maddison et al., 2017).

6.1.1 Module Controller

The proposed Transformer modular controller is based on the work on deep averaging network (DAN, Iyyer et al., 2015; Cer et al., 2018). DAN architecture presents a reasonable trade-off between the quality of the extracted sequence representations and the network complexity. Figures 6.1 and 6.2 illustrate the general controller implementation and the masking of individual modules in a modular Transformer block respectively.

[2] A network with M selectable modules has 2^M possible module combinations.

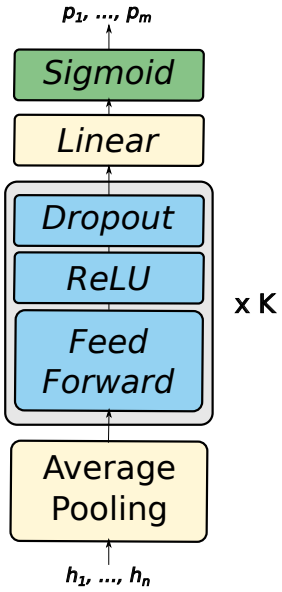

Figure 6.1: Schema of the modular controller based on deep averaging network. Controller input is processed by a series of feed-forward blocks and the output is transformed by a sigmoid non-linearity into a set of Bernoulli distributions. This figure illustrates the controller predicting a single set of masks for the whole sentence (CtrlSeq). The average pooling layer is omitted in CtrlTok and each input is processed in isolation.

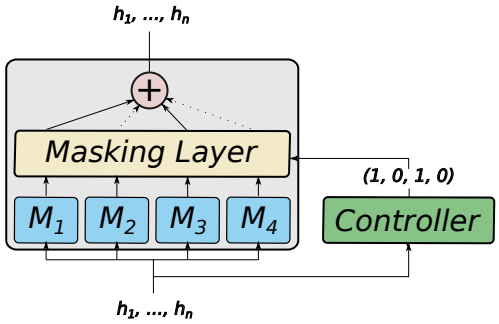

Figure 6.2: Schema of a generic modular layer. Layer input is processed by the layer modules and the controller. The controller generates a set of binary masks and disables a subset of modules (modules M_2 and M_4 in the example).

Going back to Equation 6.1, the Transformer controller takes the block input sequence $\boldsymbol{h} = (h_1, .., h_n)$, which is either the whole input sequence in the case of the encoder or the current partial hypothesis in the case of the decoder, and produces a set of masks $\boldsymbol{\xi} = \{\xi_1, .., \xi_M\}$, each mask being either 0 or 1. First, the input sequence $\boldsymbol{h}$ is transformed into a bag of words (BoW) representation using average pooling (Hubel and Wiesel, 1959; Scherer et al., 2010):

$$\bar{h}^{(0)} = \frac{1}{N} \sum_{i=1}^{N} h_i \tag{6.5}$$

The pooled sequence is then used to predict a set of masks that will be applied for the whole sequence. In the case of the decoder, in the k-th decoding step, the sequence of $(h_1, .., h_k)$ decoder states is used to predict the mask for that decoding step. We will refer to this input processing as *CtrlSeq*. Similarly to Zhang et al. (2021a), we can also replace the input pooled state $\bar{h}$ with the individual states h_i and use each state in isolation to predict its respective module mask, both in the encoder and decoder. We will refer to the token-level mask prediction as *CtrlTok*.

Using notation similar to that of Section 3.2, the controller with depth K transforms the sequence representation $\bar{h}^{(0)}$ using a set of functions $\{f'^{(0)}, .., f'^{(K-1)}\}$, each followed by dropout:

$$\bar{h}^{(j+1)} = f'^{(j)}(\bar{h}^{(j)}) \tag{6.6}$$

$$f'(\bar{h}) = \mathit{ReLU}(W'\bar{h} + b') \tag{6.7}$$

where the parameters $W' \in \mathbb{R}^{d \times d}$, $b' \in \mathbb{R}^d$ are the projection and bias parameters respectively. The output of the final controller layer $\bar{h}^{(k)}$ is then linearly transformed into M logits $l_1, .., l_M$ of size M, each logit corresponding to the respective module mask. For simplicity, we assume mutual independence between the individual masks. The controller can then generate a mask using the characteristic function $\xi_i = \mathbf{1}_{l_i > 0}$.

The characteristic function is non-differentiable and thus not suitable for backpropagation. Similarly to the work on variational autoencoders, we would also like to draw samples from the Bernoulli distribution conditioned on the logits $l_1, .., l_M$. For this reason, we replace the characteristic function with Gumbel-sigmoid (Csordás et al., 2021), a continuous relaxation of the Bernoulli distribution which can be derived from the Gumbel-softmax relaxation of the categorical distribution. Given two uniform distributions U_1 and U_2, on the interval $(0, 1)$, temperature τ and the output logit l_i, we can draw binary samples using the following function:

$$s(l_i) = \sigma\left(\frac{1}{\tau}\left(l_i - \log\frac{\log U_1}{\log U_2}\right)\right), \tag{6.8}$$

The temperature enables adjusting exploration vs exploitation: high-temperature results in a less "peaked" sigmoid, resulting in output values closer to 0.5. As τ approaches 0, the samples get closer to the samples from the discrete Bernoulli distribution. The Gumbel-sigmoid with temperature annealing can be used for a *soft* approximation of the gradients during model training, allowing updates to all modules in each training step. However, we can also use the *hard* mask values generated by the characteristic function and make the controller output differentiable using the straight-through estimator (STE, Bengio et al., 2013):

$$\xi_i = [\mathbf{1}_{s(l_i) > 0.5} - l_i] + l_i \tag{6.9}$$

where $[x]$ indicates disabled gradient flow.

Lastly, we include an additional regularization term that allows controlling the ratio of the active blocks. Originally, Zhang et al. (2021a) introduced a *budget* hyperparameter $p \in [0, 1]$, to control the gating mechanism in their adapter extension of the Transformer NMT, forcing a network to apply p language-specific and $1 - p$ language-independent adapters to a given output. We use the same regularization term to adjust the ratio of selected modules that the network is allowed to use. Having a set of M modules, for a given sequence pair $\boldsymbol{x}, \boldsymbol{y}$, let $G_i = \{\xi_1, .., \xi_m\}$ indicate a set of masks generated during the processing of the sequence pair.[3] Using that notation, we can then expand the loss function using the following budget regularization:

$$L(\boldsymbol{\theta}, \boldsymbol{\phi}) = L_{NLL}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \left| \frac{\sum_{i=1}^{M} \sum_{\xi \in G_i} \xi}{\sum_{j=1}^{M} |G_j|} - p \right| \tag{6.10}$$

Besides forcing the network to try allocating different combinations of modules (due to a high sampling temperature), in case of module collapse, i.e. the selection only of a fixed subset of modules, regardless of the input, the regularization can still help to remove unnecessary model parameters, working as a type of pruning mechanism.

6.1.2 Modular Blocks

Given the set of binary masks $\boldsymbol{\xi} = \{\xi_1, .., \xi_M\}$ produced by the module controller, the modules are selected according to Equation 6.1. More precisely, we propose the following modification to the existing FFN and MHA blocks to support module masking.

Masked multi-head attention (MMHA) is based on the MHA masking introduced by Michel et al. (2019). Although only using fixed attention masks, the authors propose the following modification to the original Equation 3.19:

$$\mathit{MMHA}(q, \boldsymbol{k}, \boldsymbol{v}) = \sum_{j=1}^{N_{head}} \xi_j \mathit{Att}_j(q, \boldsymbol{k}, \boldsymbol{v}) \tag{6.11}$$

We use the query q to predict the module masks $\boldsymbol{\xi}$.

This allows the conditioned masking of unnecessary attention heads, providing a better ground for attention head specialization. The previous work investigating attention head specialization (Voita et al., 2019; Michel et al., 2019; Mareček and Rosa, 2019) showed that some attention heads can take on certain positional, syntactic, or

[3]For decoder controllers, $|G_i| = |\boldsymbol{y}|$, for encoder, $|G_i| = |\boldsymbol{x}|$ when using CtrlTok or $|G_i| = 1$ when using CtrlSeq.

vocabulary-related roles. On the other hand, it is not clear how these specialized heads behave when the phenomena they focus on are not present in the processed input. Even in these cases, the attention head contributes to the encoding and decoding process. In theory, hard masking could help the specialization process, better allocating the network capacity. In the rest of this chapter, we will refer to attention heads (and FFN modules) as modules for simplicity.

Analogously to MMHA, we propose the masked feed-forward network (MFFN). Based on Equation 3.20, we modify the Transformer FFN in the following way (we omit biases for simplicity):

$$\mathit{MFFN}(h_i) = \sum_{j=1}^{N_{module}} \xi_j \mathit{FFN}_j(h_i) \tag{6.12}$$

The original projection matrices $W_1, W_2 \in \mathbb{R}^{d \times 4d}$ get separated into N_{module} modules, where $W_1^{(j)}, W_2^{(j)} \in \mathbb{R}^{d \times \frac{4d}{N_{module}}}$ used by their respective FFN_j block can then be independently enabled/disabled by the controller.

The main inspiration behind the MFFN is the work on Transformer adapters (Zhang et al., 2021a; Bapna and Firat, 2019; Pfeiffer et al., 2020). Contemporary research adds a feed-forward adapter layer after all or only certain Transformer FFN blocks. The previous results show that adding adapters can improve model fine-tuning and transfer learning in the multilingual setting. Due to the nearly identical nature of the standard Transformer FFN blocks and the adapter layers (both are two linear transformations with a non-linearity in-between), the combination of the functionality of the Transformer FFN block with the modularity of adapters could help reduce the overall number of Transformer parameters while borrowing the benefit of both layer types.

6.2 Experiments: String Editing

Our initial experiments investigated the ability of the modular Transformer to adapt in a simplified multi-task setting. The goal of this analysis, similar to previous chapters, was to understand the model behavior on tasks with low ambiguity before moving to a more difficult NLP task of NMT.

We used the multi-task string editing dataset with the following four operations: `push`, `pop`, `shift`, and `unshift`. Each dataset contains a randomly generated input sequence of tokens a, b with a maximum length of 25 tokens, a task prefix token that instructs the model which editing operation should be applied to the input and the

correct output sequence. We generate 30k unique examples for each operation and set aside 1,000 validation and 1,000 test examples. To test the modular Transformer in the multi-task settings, we concatenate all the remaining data from all tasks, shuffle them and use this dataset combination for training.

We use the modular Transformer that we described earlier in this chapter. The details about the basic model hyper-parameters are available in Appendix A.7. We compared multiple variations of the modular Transformer. We applied the module controller on either the MMHA blocks (*attn*), MFFN blocks (*ffn*), or a combination of both (*full*). In all cases, we introduce the controller to both the encoder and decoder, each Transformer block being assigned a separate controller network. Furthermore, we investigated the differences between the controller that processes each token in isolation to predict token-specific masks (*CtrlTok*) and the controller that predicts sequence masks using the average pooling on the whole sequence before processing it by the controller feed-forward layers (*CtrlSeq*). Lastly, we investigated the differences between the soft mask prediction, i.e. the controller generating mask values from the $(0, 1)$ interval by sampling the values from the Gumbel-sigmoid output layer (*soft samples*) and the discrete 0 or 1 mask samples created from the soft samples by the characteristic function and propagating the gradients using STE during training (*hard samples*).

We compared the model variations across several values of the budget hyper-parameter to see how much the regularization affects the resulting model accuracy and module coverage. Figure 6.3 (top) shows that the string-level accuracy (ACC, described in Section 5.3.2) is not affected by the budget, possibly due to the simplicity of the string-editing multi-task setting. Only the soft sample methods combined with the modular attention models (*attn*, *full*) suffer from the enforced budget regularization.

To further analyze the module selection mechanism, we define the *module selection probability* (or *module selection*, in short):

$$p_m = p(m = 1) = \frac{count(m = 1)}{count(m)} \tag{6.13}$$

We measure the module $count(m)$ (how many times was controller deciding to select a module) and module selection $count(m = 1)$ (how many time was the module selected by the controller) using our test dataset. Comparing the average module selection (averaged across all modularized block modules), the CtrlTok models are much better at respecting the budget criterion than the CtrlSeq (Figure 6.3, bottom). Still, in both cases, the results show that the budget regularizer can reduce the number of activated parameters when processing individual inputs.

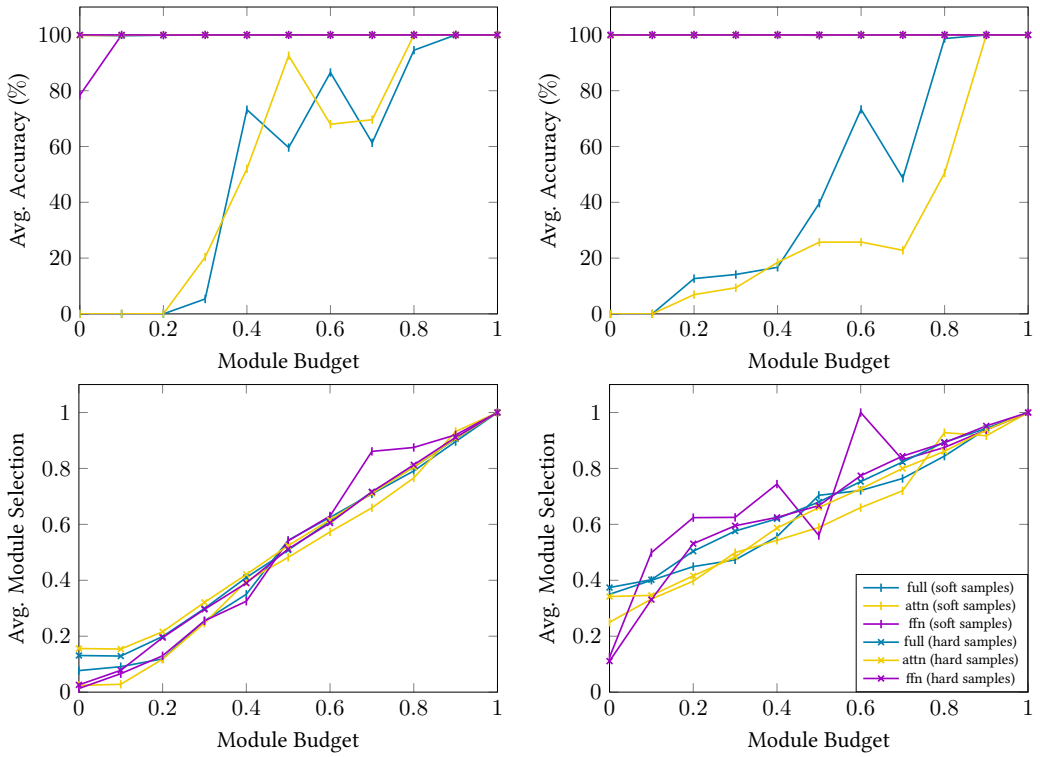

Figure 6.3: Basic model performance with respect to varying values of the budget regularizer. **Top**: Model accuracy averaged over all tasks. **Bottom**: Average mask selection (across the whole model). **Left**: Models using CtrlTok controller. **Right**: Models using CtrkSeq controller.

To further explore the properties of the pruning effect, we compared the module selection averaged across the individual Transformer block types. Figure 6.4 shows that the CtrlTok modular Transformers tend to "prune" the network more or less evenly with the decreasing module budget without a clear preference for any specific type of Transformer block (all blocks have a similar average module selection). Still, the encoder-decoder attention seems to be slightly more preferred by the model compared to the other attention blocks (Figure 6.4, top left and middle left). Since the encoder-decoder attention is, in theory, responsible for transferring the information between the encoder and decoder, this block preference is expected. However, this does not hold for the CtrlSeq models. These models consistently prefer selecting encoder-related modules, and pruning the rest of the network, if possible. We argue that this might be a result of the task-label tokens influencing all controller predictions in the encoder due to the average sequence representations used for the module prediction. The encoder module preference suggests that the information about the string operation is in this case encoded mostly by the encoder modules, possibly reducing the workload for the encoder-decoder modules.

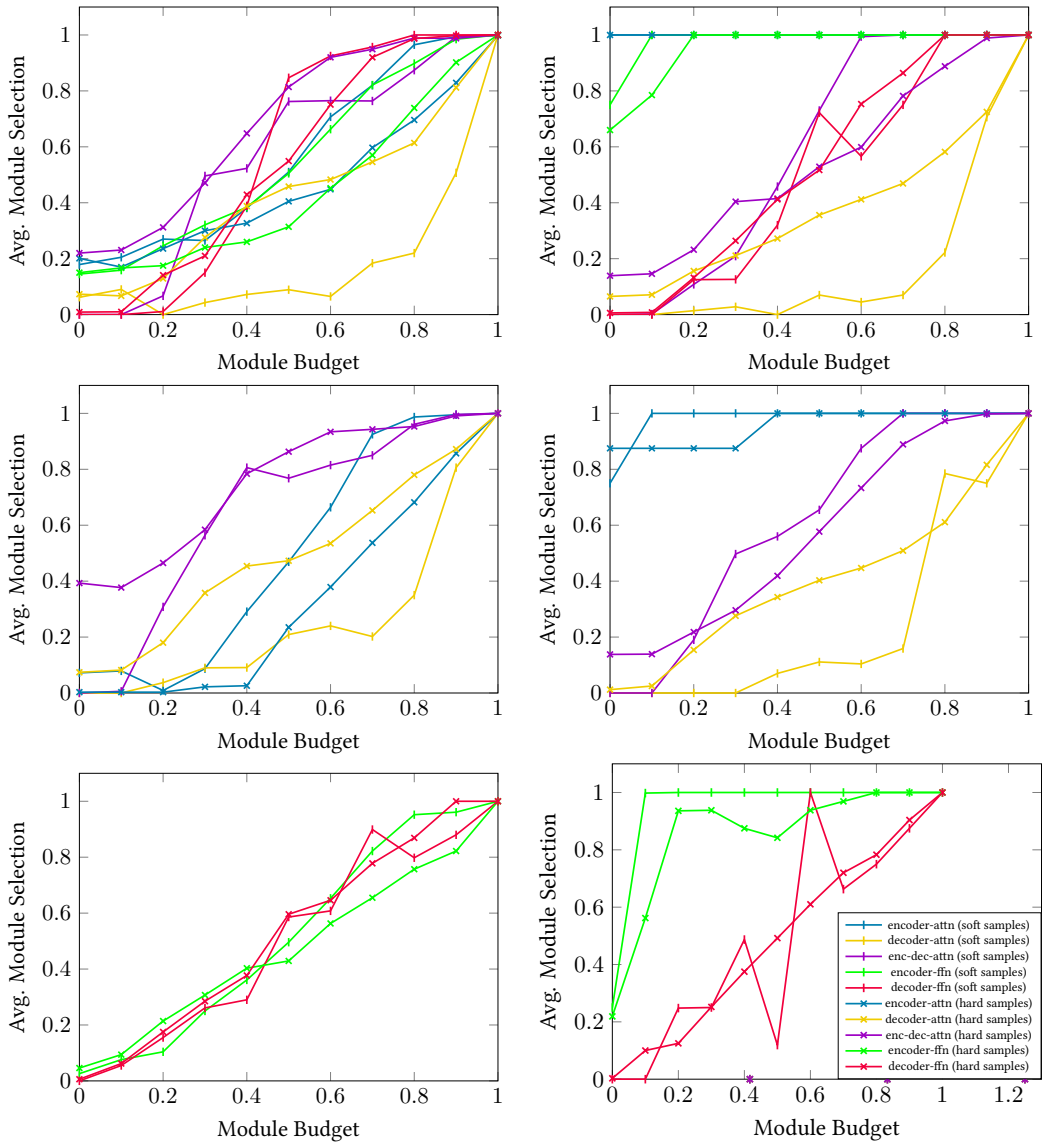

Figure 6.4: Average module selection with respect to individual types of modular blocks. The comparison of block type selection in the *full* modular Transfomer (**top**), *attn* modular Transformer (**middle**), and *ffn* modular Transfomer (**bottom**). **Left**: Mask prediction using individual tokens. **Right**: Masks predicted for the whole sequence using average pooling on the controller input. The block types (colored lines in the graph) are excluded from the subgraphs if they are not modularized in a given Transformer variant.

The results so far show a promising utility of the budget regularizer constraint, forcing the model to allocate only a limited amount of its available modules. However, such a behavior could be achieved even if the modules collapsed and the same set of modules was used throughout the test set. To see whether the module collapse is avoided in our models, we measured the average entropy of the module prediction. Using the module selection probability in Equation 6.13, we measure the module se-

lection entropy:

$$H(m) = -p_m \log p_m - (1 - p_m) \log(1 - p_m) \tag{6.14}$$

The entropy computation in Equation 6.14 is similar to the batch entropy computation proposed by Kirsch et al. (2018), however, we also use our formulation for later evaluation of individual module selection. Furthermore, our proposed entropy measure disregards the mask prediction distribution provided by the model and instead is based on the mask selection frequencies measured using the test data. Using Equation 6.14, high entropy implies that module selection is conditioned on the input while low entropy implies that a particular module mask is either 0 or 1 most of the time, regardless of the input.

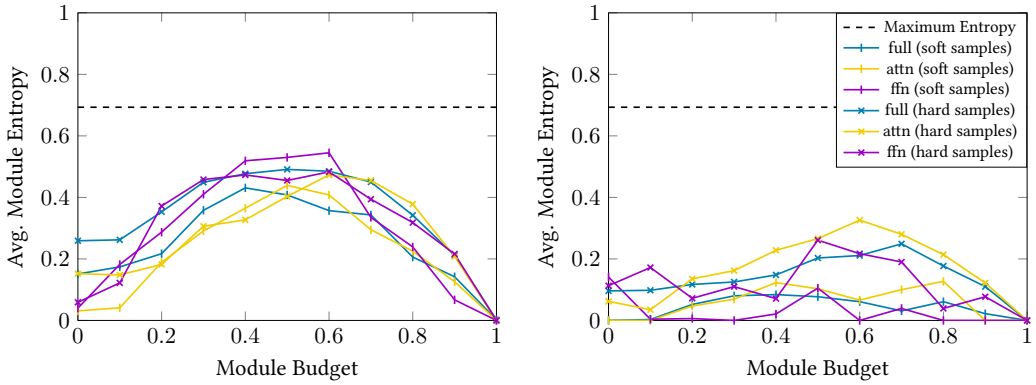

Figure 6.5: Average module selection entropy of the modular Transformer variants. High entropy hints at the module selection being strongly conditioned on the input. Low entropy implies module collapse, setting most of the module masks to either 0 or 1 most of the time. **Left**: Transformer with the token-level controller (CtrlTok); entropy averaged across all test tokens. **Right**: Transformer with the sequence-level controller (CtrlSeq); entropy averaged across all test sequences. The dashed line indicates the upper limit of the module entropy.

Figure 6.5 shows the average module selection entropy with respect to the budget regularization constraint. Regardless of the modular Transformer variant, the token-level controller network shows promising results with average entropy copying entropy implied by the constraint value (0.0 and 1.0 budget leading to low entropy of selection and 0.5 resulting in high entropy). The sequence-level controller does not perform as well, suggesting that a higher level of module collapse is taking place. Similarly to the accuracy results (Figure 6.3), the soft sampling method seems to be less effective, in this case, is more prone to module collapse.

To conclude the multi-task string editing experiments, we measured whether the modules specialize with respect to individual string-editing tasks. Given a random variable t representing one of the four tasks, we measure the conditional entropy of the task given the module mask value m:

$$H(t|m) = -\sum_{t,m \in \boldsymbol{D_{test}}} p(t,m) \log \frac{p(t,m)}{p(m)} \tag{6.15}$$

Due to similar sizes of the testsets for the individual tasks, the value probability $p(t)$ is roughly 0.25 for each of the four tasks. Given Equation 6.15, low conditional entropy implies that the information about a module selection can help us identify which task (or a subset of tasks) is being processed. This, consequently, suggests that the module is specialized for some of the tasks. High conditional entropy implies the opposite: the module does not give as much information about specific tasks and is, therefore, task invariant.

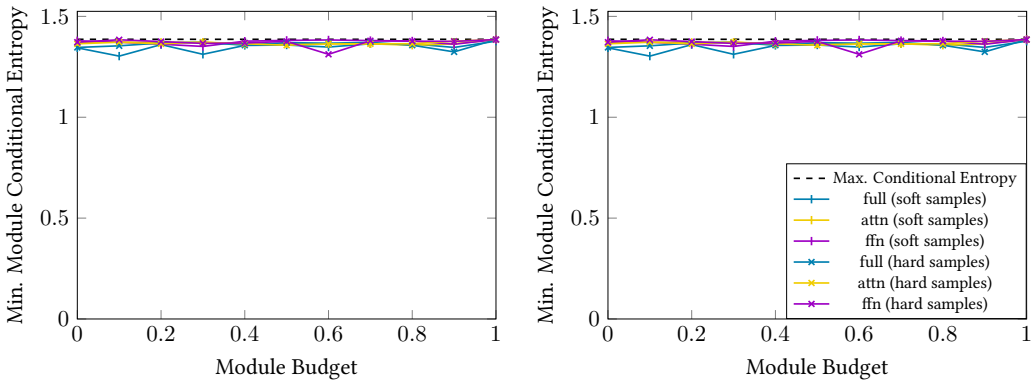

Figure 6.6: Task-conditional entropy of to the lowest conditional entropy modules in the respective modular Transformer variants. Low entropy implies task-conditioned selection of a module, i.e. specialization of the module, high entropy implies the opposite. **Left**: Transformer with the token-level controller (CtrlTok). **Right**: Transformer with the sequence-level controller (CtrlSeq). The dashed line indicates the upper limit of the task-conditional entropy.

Figure 6.6 shows the task-conditional entropy of the modules with the lowest conditional entropy. Contrary to our hypothesis, the observed high entropy suggests that no task specialization occurs in any of the Transformer modules. A possible explanation behind the lack of task specialization would be that the task-related information signal is too weak for the controller to focus on this signal. Thus, the module specialization indicated by the measuring of module selection entropy is likely bound by different criteria.

6.3 Experiments: Multilingual NMT

Lastly, we investigate the modular Transformer on the multilingual NMT task. Compared to string editing, individual tasks (translation language pairs) are not as distinguished and the model can, in theory, benefit from the task overlap (e.g. when learning to translate from various languages into English).

We use the same OPUS-100 dataset and preprocessing described in Section 5.3.3. We use the same bilingual and jointly-trained, multilingual baselines for comparison with our modular Transformer multilingual models. The details about the modular Transformer hyper-parameters are described in Appendix A.8. Similarly to the string editing experiments, we compare the fully modular Transformers (*full*) with the variations that have either only modular attention (*attn*) or modular FFN blocks (*ffn*). Furthermore, we provide a comparison between the token-level and sequence-level mask prediction. Due to the consistently poorer performance of the soft sampling approach in the previous section, we only use the hard sampling method.

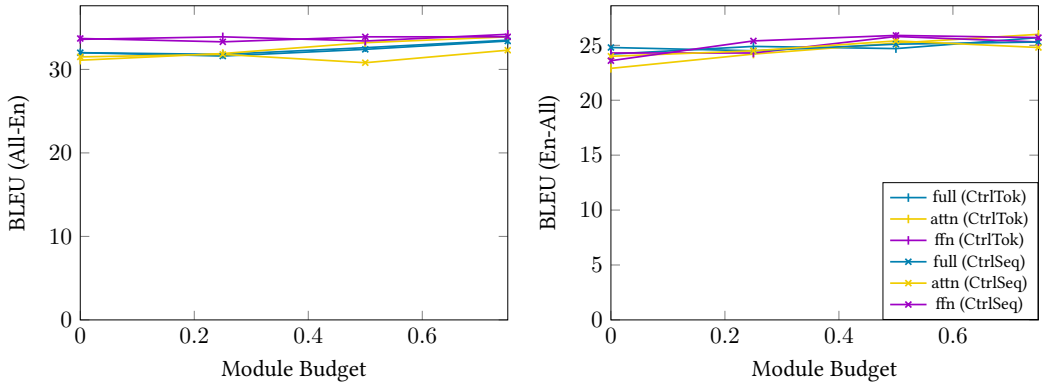

Figure 6.7: Model performance of modular multilingual Transformers with respect to different values of the budget hyper-parameter p. **Left**: Many-to-one models. **Right**: One-to-many models.

We compare the selected models on many-to-one, one-to-many, and many-to-many machine translation. Again, the models translating into multiple languages receive a special language token prefix to the input sentence to indicate the target language of the reference translation. We train each model for 600k updates and use the checkpoint with the best performance on the validation data for the final evaluation.

We measure the performance of the models with budget $p \in \{0.0, 0.25, 0.5, 0.75\}$.[4] Figure 6.7 shows the performance comparison of the many-to-one and one-to-many modular Transformer variants with respect to values of the budget hyper-parameter. The performance was computed using a concatenation of all the respective language pair testsets. Even though the model performance mostly drops with the decreasing value of p, the drop in BLEU does not seem too significant.

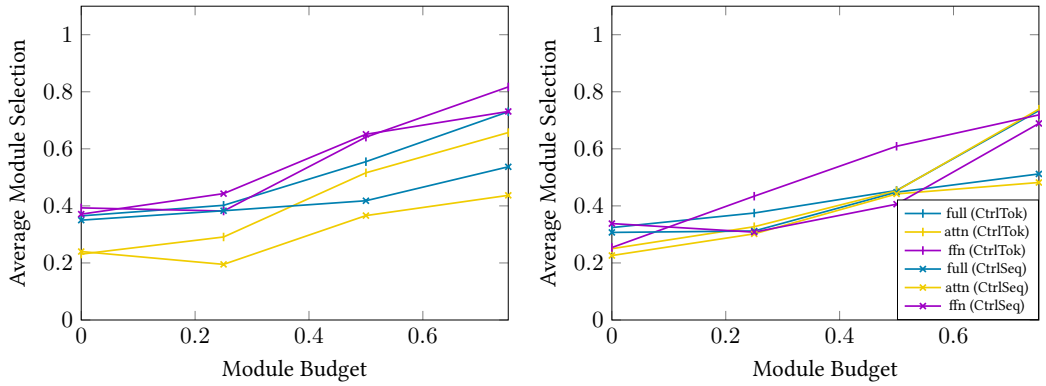

Figure 6.8: Average module selection in modular multilingual Transformers with respect to different values of the budget hyper-parameter p. **Left**: Many-to-one models. **Right**: One-to-many models.

Although the differences in BLEU are not too large, the controller still tries to satisfy the budget constraint (Figure 6.8). Compared to the string editing experiments, the low-budget settings result in a higher average module selection, hinting at the higher capacity required by the NMT modular Transformer. Still, a fair amount of pruning is performed, selecting on average less than half of the modules at the lower budget values.

To perform a more detailed model comparison of the modular Transformer variants, we select the budget with the best validation BLEU. Following the initial probing of the modular multilingual models, we compared the modularized models with their standard bilingual and multilingual counterparts in terms of translation quality. Besides BLEU, we also use COMET (Rei et al., 2020), a metric using a Transformer model optimized for automatic evaluation machine translation. COMET has a reportedly better correlation with human judgements than the standard n-gram precision metrics. We use both the version with reference (Cmt) and the reference-less version of the metric (Cmt_{qe}).

[4]We are not interested in the model budget 1.0, i.e. modularization with the full set of modules at all times.

ID	System	De-En			Zh-En			Br-En			Te-En		
		BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}
1	base De→En	32.5	0.2374	0.1107	–	–	–	–	–	–	–	–	–
2	base Zh→En	–	–	–	38.9	0.3234	0.1143	–	–	–	–	–	–
3	base Br→En	–	–	–	–	–	–	15.9	-0.3841	0.0973	–	–	–
4	base Te→En	–	–	–	–	–	–	–	–	–	24.8	0.0372	0.1029
5	base All→En	31.1	0.2266	0.1117	39.1	0.3370	0.1149	18.3	-0.2328	0.0995	30.2	0.2121	0.1059
6	CtrlTok-full-0.75 All→En	31.6	0.2228	0.1109	37.9	0.3190	0.1134	**18.8**	**-0.1922**	**0.0987**	29.7	0.1947	0.1047
7	CtrlTok-ffn-0.25 All→En	31.6	0.2190	0.1109	38.6	0.3359	0.1143	**19.6**	**-0.1635**	**0.0993**	30.0	**0.2219**	0.1057
8	CtrlTok-attn-0.75 All→En	**32.0**	**0.2366**	0.1109	38.2	0.3119	0.1131	**19.7**	**-0.1792**	**0.0986**	30.1	**0.2325**	0.1048
9	CtrlSeq-full-0.75 All→En	30.0	0.1672	0.1091	36.0	0.2603	0.1115	17.0	-0.2718	0.0978	25.5	0.0191	0.1043
10	CtrlSeq-ffn-0.5 All→En	**31.6**	**0.2316**	0.1110	38.6	0.3224	0.1139	**18.6**	**-0.2104**	0.0986	30.3	**0.2305**	0.1055
11	CtrlSeq-attn-0.75 All→En	30.7	0.1844	0.1098	37.0	0.2791	0.1120	17.0	-0.3106	0.0973	27.2	0.1427	0.1048
12	base All→All	28.7	0.0922	0.1080	35.3	0.2314	0.1122	14.4	-0.4291	0.1004	20.8	-0.0879	0.1066
13	CtrlTok-full-0.75 All→All	**29.5**	**0.0987**	0.1074	34.6	0.2224	0.1110	**15.7**	**-0.3853**	0.0999	**22.6**	**-0.0281**	0.1055
14	CtrlTok-ffn-0.5 All→All	27.9	0.0536	0.1070	33.8	0.2064	0.1120	**15.1**	**-0.4144**	0.1004	20.4	**-0.0788**	0.1054
15	CtrlTok-attn-0.5 All→All	**29.2**	**0.1002**	0.1075	34.5	0.2256	0.1112	**15.1**	**-0.3900**	0.0997	**24.4**	**-0.0058**	0.1061
16	CtrlSeq-full-0.5 All→All	27.9	0.0343	0.1055	32.7	0.1542	0.1096	**15.3**	**-0.4018**	0.0986	19.6	-0.1370	0.1039
17	CtrlSeq-ffn-0.5 All→All	28.1	0.0810	0.1070	34.0	0.2113	0.1118	**15.4**	**-0.4088**	0.1002	**21.2**	**-0.0764**	0.1056
18	CtrlSeq-attn-0.5 All→All	28.1	0.0504	0.1060	33.2	0.1733	0.1097	14.4	-0.4654	0.0992	20.1	-0.1222	0.1061

Table 6.1: Model comparison on many-to-one translation. We compare both many-to-one and many-to-many model variants. Scores of modular models that outperform their respective non-modular variants are highlighted in **bold**. Although we do not focus on a direct comparison with the bilingual baselines, we include them for reference.

ID	System	En-De			En-Zh			En-Br			En-Te		
		BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}	BLEU	Cmt	Cmt_{QE}
1	base En-De	29.5	0.1600	0.1090	–	–	–	–	–	–	–	–	–
2	base En-Zh	–	–	–	26.1	0.2342	0.1113	–	–	–	–	–	–
3	base En-Br	–	–	–	–	–	–	17.3	-0.4024	0.0907	–	–	–
4	base En-Te	–	–	–	–	–	–	–	–	–	20.3	0.0963	0.0973
5	base En-All	28.4	0.1189	0.1081	29.5	0.2806	0.1116	17.5	-0.3121	0.0913	22.0	0.2723	0.0983
6	CtrlTok-full-0.75 En→All	28.4	0.1021	0.1074	**31.3**	0.2677	0.1105	17.5	-0.2808	0.0911	**22.5**	0.2726	0.0980
7	CtrlTok-ffn-0.5 En→All	28.6	0.1099	0.1084	29.7	0.2810	0.1113	**18.3**	-0.2687	0.0915	**22.9**	0.2773	0.0983
8	CtrlTok-attn-0.75 En→All	**29.3**	0.1321	0.1079	28.5	0.2699	0.1109	**18.8**	-0.2577	0.0913	21.1	0.2700	0.0983
9	CtrlSeq-full-0.75 En→All	27.9	0.0910	0.1068	28.7	0.2222	0.1099	17.8	-0.3011	0.0917	21.9	0.2577	0.0980
10	CtrlSeq-ffn-0.5 En→All	28.6	0.1271	0.1075	29.0	0.2664	0.1107	**18.9**	-0.2598	0.0918	**22.8**	0.2934	0.0992
11	CtrlSeq-attn-0.5 En→All	28.7	0.1061	0.1068	**31.0**	0.2571	0.1098	16.5	-0.3425	0.0909	20.5	0.2289	0.0986
12	base All→All	26.0	0.0382	0.1061	28.3	0.2165	0.1097	14.0	-0.4271	0.0913	17.2	0.472	0.0964
13	CtrlTok-full-0.75 All→All	26.2	0.0054	0.1049	28.4	0.2087	0.1089	**14.8**	-0.4063	0.0913	18.0	0.1471	0.0975
14	CtrlTok-ffn-0.75 All→All	26.2	0.0283	0.1055	25.6	0.2048	0.1093	**14.6**	-0.4338	0.0913	17.0	0.0171	0.0965
15	CtrlTok-attn-0.75 All→All	26.1	0.0181	0.1054	28.3	0.2158	0.1093	**15.9**	-0.3815	0.0905	18.2	0.1177	0.0967
16	CtrlSeq-full-0.75 All→All	25.9	-0.0195	0.1039	26.8	0.1684	0.1077	**15.3**	-0.3852	0.0918	**17.8**	0.0582	0.0961
17	CtrlSeq-ffn-0.5 All→All	26.0	0.0188	0.1054	28.2	0.2050	0.1094	14.3	-0.4199	0.0915	**18.4**	0.1420	0.0973
18	CtrlSeq-attn-0.75 All→All	25.9	-0.0216	0.1045	27.3	0.1740	0.1083	14.4	-0.4201	0.0904	**17.7**	0.0919	0.0967

Table 6.2: Model comparison on one-to-many translation. We compare both one-to-many and many-to-many model variants. Scores of modular models that outperform their respective non-modular variants are highlighted in **bold**. Although we do not focus on a direct comparison with the bilingual baselines, we include them for reference.

Tables 6.1 and 6.2 show the comparison of the many-to-one and one-to-many models. As expected the many-to-one and one-to-many models outperform more complex many-to-many models. When translating to English (Table 6.1), the multilingual models can exploit the additional English data provided with the high-

resource languages (German, Chinese) to improve the low-resource translation (Breton, Telugu).[5] Compared to the standard Transformer, CtrlTok models improved the translation from both German and Breton without losing performance on the other two languages, suggesting better utilization of the combined parallel training data. This trend can be seen even on many-to-many translations with additional improvement in Telugu and in opposite translation direction (Table 6.2).

In one-to-many translation (Table 6.2), the improvements are not as large as in the other directions. Still, our models were able to beat the multilingual baseline in either Chinese (*full* and *attn* settings) or German and Telugu (*attn*). Overall, we conclude that the modular Transformers benefited mostly from the increase of the target-side data, making better use of the many-to-English datasets in a many-to-one direction, possibly due to the selective nature of the controller, which better distributes the knowledge in the training data. It is not clear from the results which model setting would be consistently the best performing. However, it seems plausible that the attention controller benefits more from the token-level mask prediction while the feed-forward modular blocks get higher gains from the sequence-level mask prediction.

The very low scores of the Comet metric, sometimes negative, are most likely a result of the validation data in OPUS-100 sampled from the original Opus corpora. The references in the sampled testsets are more likely to be a result of automatic sentence alignment rather than genuine human translations. For this reason, we also did a small-scale manual evaluation within the confines of our available resources. Based on the results in Table 6.1, we compared the baseline model in row (5) and a modular variant in row (8) with their German and Chinese bilingual counterparts (row (1) and (2), respectively). We could not the perform manual evaluation of the low-resource languages due to the inavailability of speakers of Breton and Telugu, although, that evaluation would have been more interesting.

For German, we used 3 distinct annotators to annotate a total of 163 unique sentences.[6] Each annotator was a second language learner of both German and an English with higher proficiency in English. For Chinese, we used a single native Chinese speaker (and English second language learner) to annotate a sample of 84 sentences. For both language pairs, each annotator was presented with the input sentence and three possible translations. The annotators were instructed to indicate which trans-

[5]Still, it is important to note that improvements of less than 1 point BLEU cannot be considered significant without a more thorough statistical significance testing.

[6]There was a partial overlap between the sentences annotated by the individual annotators.

lations were good (∗), very good (∗∗), or bad (×) according to their judgement. Due to the nature of the OPUS-100 dataset (reference sentences are not necessarily manual translations of the source), we did not present the annotators with the reference translation to avoid introducing bias to their annotations.

	∗∗	∗	×
base de-en (1)	7	92	51
base all-en (5)	8	76	61
modular all-en (8)	9	79	60

Table 6.3: Results of manual comparison of the baseline bilingual and multi-lingual NMT systems with the selected modular Transformer on German-to-English translation. The ∗∗, ∗ and × indicate how many times the produced translation was very good, good, and bad, respectively, according to the annotators.

	∗∗	∗	×
base zh-en (2)	4	22	13
base all-en (5)	7	23	14
modular all-en (8)	5	24	12

Table 6.4: Results of manual comparison of the baseline bilingual and multi-lingual NMT systems with the selected modular Transformer on Chinese-to-English translation. The ∗∗, ∗ and × indicate how many times the produced translation was very good, good, and bad, respectively, according to the annotators.

Tables 6.3 and 6.4 present the results of both German and English evaluation respectively. Both multilingual systems produce slightly more *bad* translations than the bilingual counterpart when translating from German while being at a similar level in this regard when translating from Chinese. Similarly, they are also outperformed in the terms of good translations by the bilingual system in German-English, being marked as good or very good only 84 and 88 times against the 99 good or very good translations of the bilingual system and outperforming the Chinese-English bilingual system with 30 and 29 good translations against 26 respectively. This more or less reflects the model comparison based on the automatic metrics supporting the results presented earlier in Tables 6.1 and 6.2. The only surprising result is a slightly better performance of the modular Transformer in Chinese-English when compared to the bilingual system which is opposite to the automatic evaluation results.

Lastly, we inspected the module selection mechanism in the *full* modular Transformers. For the following analysis, we used the model in row (6) from Tables 6.1 and 6.2 to investigate the masking mechanism with respect to any Transformer block.

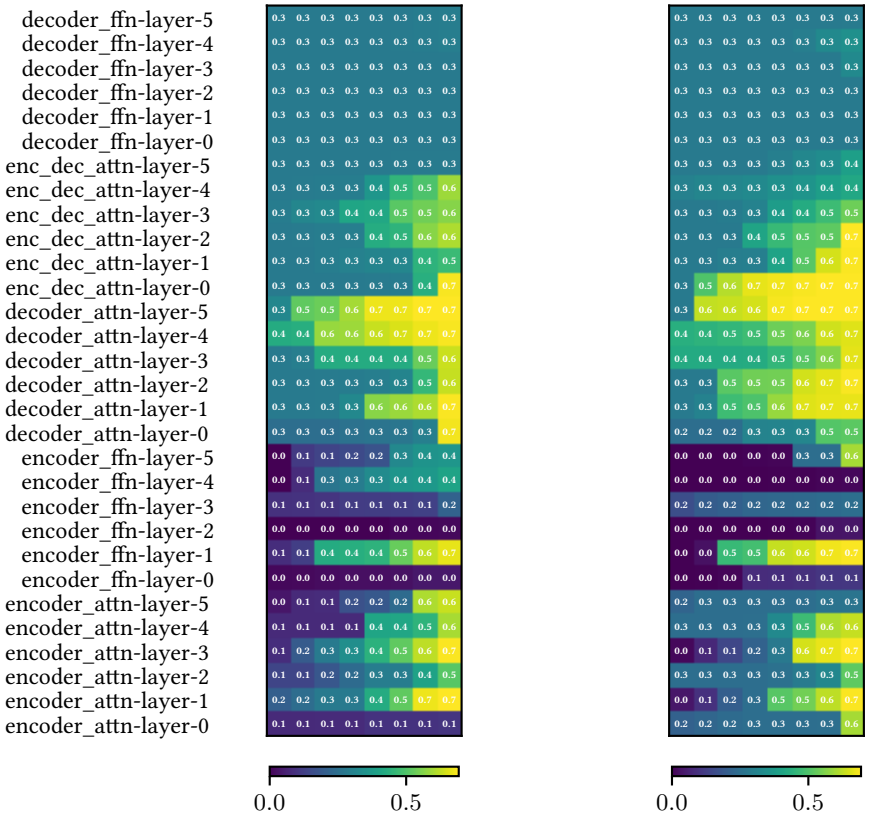

Figure 6.9: Individual module entropies of the *full* modular Transformer measured using the combined multilingual testset. The values in each layer (*encoder_attn*, *encoder_ffn*, *decoder_attn*, *enc_dec_attn*, *decoder_ffn*) are sorted in the increasing order. The higher the entropy of a particular module, the more the selection of that module depends on a specific input. **Left**: One-to-many model. **Right**: Many-to-one model. As in Figure 6.10, the order of the layers does not directly reflect the order of processing.

Figure 6.9 shows the entropies of the individual modules with respect to each Transformer layer. The majority of the high-entropy modules is located in the encoder and decoder self-attention, slightly less in the encoder-decoder attention. The implied conditional selection of these modules suggests higher module specialization in these layers. We conclude that this positively supports previous findings about attention module (head) specialization in Transformers (Voita et al., 2019). More modules with high selection entropy are located in the many-to-one model. We think that this is the result of the many-to-one source-side inputs being more diverse thus conditioning the controller to be more selective. Furthermore, this also suggests

that the language-related token signal in one-to-many models is not strong enough (less high-entropy modules). Except for a few modules in the second encoder FFN block, the FFN module specialization is generally lower, although, still happening ($entropy > 0.3$).

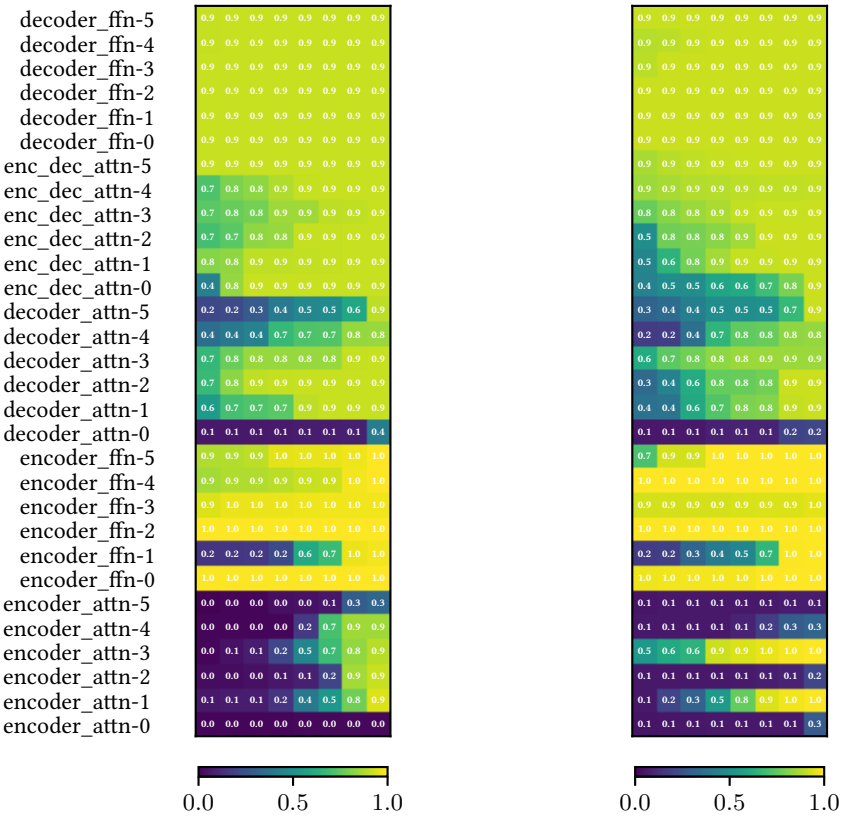

Figure 6.10: Individual module selection probabilities of the *full* modular Transformer measured using selection frequencies in the combined testset. The values in each layer (*encoder_attn*, *encoder_ffn*, *decoder_attn*, *enc_dec_attn*, *decoder_ffn*) are sorted in the increasing order. **Left**: One-to-many model. **Right**: Many-to-one model. The order of the layers does not directly reflect the order of processing.

Low entropy can imply both a high or low selection of modules. To get more information about the FFN module selection behavior, we inspected the average selection rate for each module (Figure 6.10). Note that the order of modules in the individual layers is different from Figure 6.9. Interestingly, most of the module pruning is focused on the encoder attention and the first layer of the decoder attention. This is opposite to the string editing multi-task experiment where the module selection rate in the encoder was slightly higher than in the decoder, even in the CtrlTok settings.

Also, more generally speaking, pruning and specialization rate seems to be related (layers with more high selection entropy modules also contain different modules with low selection rate). It is interesting to see different module distribution strategies in the encoder between the one-to-many (selecting a lesser amount of modules in more layers) and the many-to-one (selecting modules mainly in layers 1 and 3) model. However, it is not clear to us at the moment whether this behavior is dataset-related or a result of randomness in training, e.g. model initialization, or module sampling.

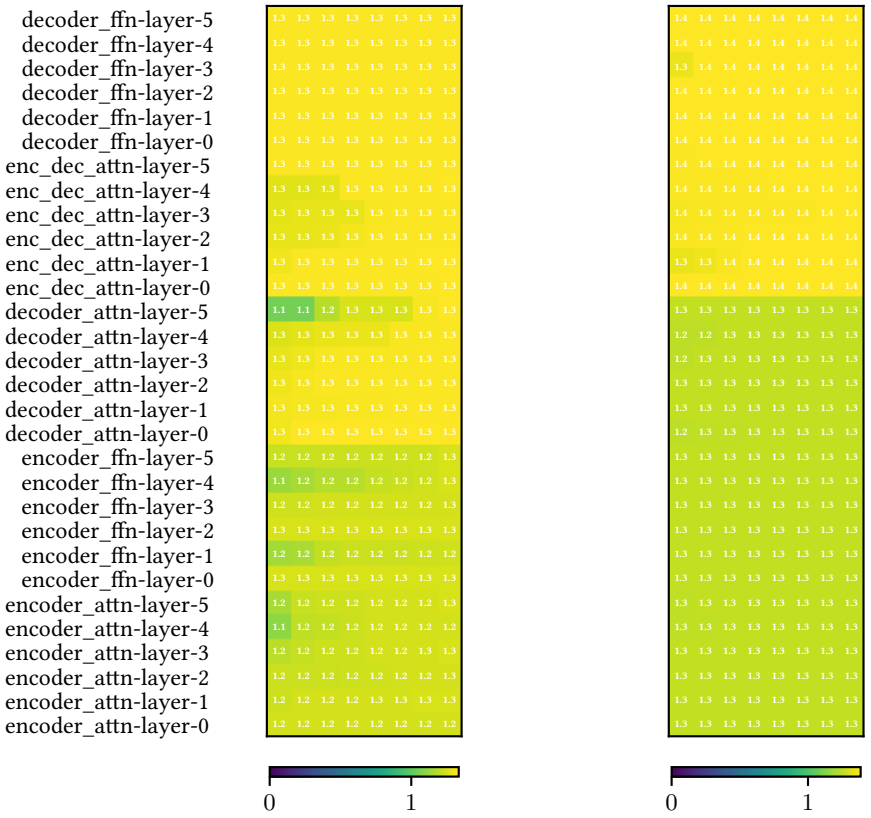

Figure 6.11: Task-conditional entropies of the *full* modular Transformer measured using the combined multilingual testset. The values in each layer (*encoder_attn*, *encoder_ffn*, *decoder_attn*, *enc_dec_attn*, *decoder_ffn*) are sorted in the increasing order. **Left**: one-to-many model. **Right**: many-to-one model. As in Figure 6.10, the order of the layers does not directly reflect the order of processing.

Lastly, we would like to see whether the modular Transformer learns to assign certain modules to a subset of tasks, making them specialized for these tasks (in this case, language-specific). Figure 6.11 shows similar results that we measured on the string editing multi-task problem. All modules have a very high task-conditional entropy with only small deviations from the average, confirming that no task (language) specialization occurs in the proposed modular Transformer. Regardless of direct

(target-language token in the one-to-many model) or indirect (different source-side languages in the many-to-one model) task indication, the module controller focuses on different aspects of the data. We leave the further investigation of the module specialization for future work.

6.4 Conclusions

We introduce a modular extension to the Transformer network enabling module masking based on the processed input. Using our implementation of a modular Transformer and the experiments analyzing it, we answer **Research Question 3:** *Can Transformer benefit from the inclusion of conditional computation?* We broke down the question into four sub-questions. We compared the proposed modular Transformer with the original Transformer implementation and using the results of our experiments, we present the following answers to these sub-questions.

RQ3.1 *What conditional computing approaches are suitable for Transformer modularization?*

We showed that mask prediction using sampling from the Gumbel-sigmoid, discretized with STE is a promising method for the modularization of the Transformer. By using temperature annealing, we were able to prevent module collapse and the introduction of budget regularization can effectively reduce the portion of modules selected by the controller. The string-editing experiments showed that sampling from the Gumbel-sigmoid distribution can be insufficient, however, we were able to improve the controller output layer by applying STE on the sampled mask probabilities.

RQ3.2 *Can the Transformer modularization lead to the effective reduction of the active model parameters?*

Our experiments have demonstrated that through training regularization, we can influence the ratio of masked modules, effectively pruning the Transformer network. The experiment results showed that this pruning leads only to a marginal drop in model performance when using STE "hard samples" during training. Later the multilingual experiment analysis showed a very systematic approach to pruning of the encoder block. This behavior could be exploited in the future for studying parameter pruning.

RQ3.3 *Does Transformer modularization lead to task specialization of individual modules?*

Measuring the task-conditional entropy of models trained on either multi-task string editing or multilingual translation suggests that modular Transformer modules do not specialize with respect to individual tasks. We think that the main reason behind the lack of task specialization is the inadequate input of information about individual tasks. Even though the prefix-based notion of tasks is effective for learning multiple tasks with different outputs, the signal from the prefix token is not strong enough to be registered by the controller.

RQ3.4 *Which Transformer blocks are better suited for modularization?*

Based on the multilingual Translation results, there is no clear advantage in choosing between the modular MHA and FFN blocks. The fully modular Transformer performed only slightly worse than the MHA and FFN variations of the architecture. The results of the multilingual evaluation show that even a combination of different blocks and token/sequence level controllers yield interesting results in the future. Even though the early encoder layers get mostly pruned by the controller, this pruning mechanism can be useful in the future for reducing the size of the model or measuring the remaining capacity of the optimized Transformer.

7
Conclusions

This chapter concludes our thesis by summarizing the main research findings. In addition, we discuss the ideas for the possible research directions for future work.

7.1 Main Findings

We focused on the analysis of the learning capabilities in Transformers. We investigated multiple aspects of Transformer learning, separating the thesis into three main topics: generalization, incremental learning, and network modularization. This section revisits the three main research questions and provides a summary of the main findings for each question.

1. **Research Question 1**: *What is the extent of the generalization ability of the current Transformer models?*

   Our experiments demonstrated severe overfitting in the Transformer with respect to the the target-side sequence lengths in the training data. We showed that one of the reasons behind this behavior is the inability of the Transformer to terminate sequence generation (by generating end-of-sequence tokens) at positions that did not contain sequence termination tokens during training. We learned using adversarial evaluation that, contrary to our hypothesis, Transformers do not exploit vocabulary distribution similarities between the training and validation data. Lastly, our rare word translation experiment showed poor generalization of the copy operation in Transformer, which is one of the applicable rules for translating unseen named-entities without prior knowledge about them.

2. **Research Question 2**: *Can selective parameter regularization lead to improvements in Transformer performance in incremental learning?*

   We showed in the unsupervised pretraining experiments, that partial regularization of the Transformer decoder using EWC can lead to slightly better translation performance on low-resource translation, compared to the regularization using language model objective. Additionally, EWC regularization results in faster training due to simpler underlying computation. The following incremental learning experiments (string editing and multilingual translation) suggested that while EWC is not able to completely remove catastrophic forgetting, it provides a mechanism for managing the trade-off between the knowledge about the original and the new task. Still, it is worth noting that the regularized NMT multilingual models were able to compete with their counterparts trained only using low-resource datasets. The results of our experiments support the conclusion from the previous work showing a similar trade-off in the area of domain adaptation in NMT (Thompson et al., 2019).

3. **Research Question 3**: *Can Transformer benefit from the inclusion of conditional computation?*

   We proposed a modularization of the original Transformer network introducing a partial masking mechanism to both attention and feed-forward Transformer blocks and extended the architecture by a controller subnetwork that uses the Gumbel-sigmoid sampling with a straight-through estimator to predict module masks based on the block input. We showed empirically that our method can provide a diverse selection of modules, being able to avoid module collapse. The controller in modular Transformer works both as a better mechanism to distribute knowledge about multiple tasks, showing slight improvements in multilingual translation, and as a pruning mechanism, reducing the number of modules required to process data for a given task. The entropy-based metric showed that modularization of the Transformer can result in module specialization, although, the modules do not specialize with respect to the different tasks.

7.2 Future Work

We suggest the following four lines of future research rooted in the answers provided by this thesis:

- *Improving the Transformer with respect to its current generalization shortcomings.*

We demonstrated the length-based overfitting of the original Transformer and its possible relation to the absolute position encoding. In the future, we suggest investigating whether the recently proposed Transformer variants that include additional position information (e.g. relative position encoding) help to alleviate the overfitting problem. Regarding named-entity translation, we hope to extend the current research results by conducting a wider study, focusing not only on the copying aspect but also on the problem of transcription between languages using different scripts. In languages such as Japanese, there are deterministic rules for transcribing foreign named-entities and loan words that a system with a good generalization ability should be able to infer.

- *Analyzing the (modular) Transformer with respect to the module specialization.*

  Even though there was no proof of task specialization in modular Transformers, we demonstrated a level of specialization in the Transformer modules. In the future, we suggest using the module selection as a form of unsupervised clustering algorithm to analyze various attributes of the dataset clusters created by the controller module selection. We believe that this cluster analysis could also help us to better understand the original Transformer architecture.

- *Applying the modular Transformer in incremental learning.*

  Although the elastic weight consolidation did not perform well on the incremental multilingual NMT, it would be interesting to investigate other approaches and whether they can be effectively combined with the modular Transformer. The resource allocation provided by the controller mechanism, in combination with methods to avoid catastrophic forgetting could lead to better utilization of the Transformer capacity in the incremental learning scenarios.

- *Studying compositionality in modular Transformers.*

  In the multi-task learning scenarios, we focused mostly on learning multiple tasks that were rather isolated, meaning that the knowledge between the tasks is shared in terms of optimizing for a shared training objective. The standard neural networks do not contain any mechanism for the explicit combination of different types of knowledge, however, the ability of a modular Transformer to select different subsets of its network could be a good stepping stone towards combining this varying knowledge. In the near future, the development of modular Transformers with respect to the zero-shot translation problem can be an interesting way of exploring the compositionality in the modular models.

A

Model Details

This section provides details regarding the model hyperparameters in the individual experiments. The models are implemented in Fairseq (Ott et al., 2019)[1], our contributions to the EWC and modular experiments are implemented in separate branches.[2,3] We use a word-level negative log-likelihood training objective with teacher-forcing (Bahdanau et al., 2014; Vaswani et al., 2017). If not stated otherwise, we use beam search decoding with beam size 4 and length penalty 0.6 during model inference. Also by default, the models share the encoder/decoder vocabulary (and embeddings) and we tie the decoder embedding matrix with the output matrix.

A.1 Sequence Length Overfitting: String Editing

We used `transformer` architecture with the following non-default model hyperparameters:

- embeddings size: 128,
- feed-forward size: 512,
- attention heads 8,
- encoder/decoder depth: 1,
- batch size: 4,096 tokens/batch,
- learning rate: 5e-4,
- label smoothing: 0.2,
- gradient clip norm: 1.0,

[1]Original Fairseq: `https://github.com/facebookresearch/fairseq`

[2]EWC implementation: `https://github.com/varisd/fairseq/tree/ewc`

[3]Modular Transformer implementation: `https://github.com/varisd/fairseq/tree/masked_modular`

- warmup steps: 4,000,
- dropout: 0.3

Each model training was terminated after 100 epochs. We use the final training checkpoint during the evaluation.

A.2 Sequence Length Overfitting: Machine Translation

We used `transformer` architecture with the following non-default model hyper-parameters:

- embeddings size: 512,
- feed-forward size: 2048,
- attention heads 8,
- encoder/decoder depth: 6,
- batch size: 4,096 tokens/batch,
- learning rate: 5e-4,
- label smoothing: 0.2,
- gradient clip norm: 1.0,
- warmup steps: 4,000,
- dropout: 0.3

We used early stopping during training: if the model performance, measured with BLEU on the complete validation dataset without the length-based splits, did not improve for 10 epochs, the training was terminated. We used the final training checkpoint during the evaluation.

A.3 Exploiting Word Distribution Similarity And Rare Word Transcription

We used `transformer` architecture with the following non-default model hyper-parameters:

- embeddings size: 512,
- feed-forward size: 2048,
- attention heads 8,
- encoder/decoder depth: 6,
- batch size: 4,096 tokens/batch,

- learning rate: 5e-4,
- label smoothing: 0.2,
- gradient clip norm: 1.0,
- warmup steps: 4,000,
- dropout: 0.3

Due to the dataset size differences, we terminate each model training after 300k parameter updates. We used the final training checkpoint during the evaluation.

A.4 Incremental Learning: Unsupervised Low-Resource NMT Pretraining

These early experiments were implemented in Neural Monkey (Helcl and Libovický, 2017).[4] We used the `transformer` architecture with the following non-default hyper-parameter values:

- embedding size: 512,
- feed-forward size: 2048,
- attention heads: 16,
- encoder/decoder depth: 6,
- batch size: 2,048 tokens/batch,
- learning rate: 3.1,[5]
- label smoothing: 0.2,
- gradient clip norm: 1.0,
- warmup steps: 33,500,
- dropout: 0.1,
- EWC λ: 0.02,

The model training is terminated using early-stopping using bilingual evaluation understudy (BLEU) scores computed using validation data. We apply beam search decoding with beam size 8 and length normalization 1.0 during model inference.

A.5 Incremental Learning: String Editing

We used `transformer` architecture with our EWC regularizer implementation with the following non-default model hyper-parameters:

[4]Neural Monkey implementation: `https://github.com/ufal/neuralmonkey`.

[5]Neural Monkey contains a different learning rate normalization resulting in different manginutes of learning rates compatible with the optimization algoritms.

- embedding size: 128,
- feed-forward size: 512,
- attention heads: 4,
- encoder/decoder depth: 1,
- batch size: 4,096 tokens/batch,
- learning rate: 0.1,
- label smoothing: 0.2,
- gradient clip norm: 1.0,
- warmup steps: 4,000,
- dropout: 0.3,

We replaced Adam optimizer (Kingma and Ba, 2014) with SGD (Robbins and Monro, 1951) in the string editing experiments for easier analysis of the network dynamics. This required changing the initial learning rate to 0.1. We trained the first task for 100 epochs and used the checkpoint with lowest validation Translation Error Rate (TER) to initialize training on the second task for additional 100 epochs. We used the checkpoint with lowest validation TER on the concatenation of the validation datasets for the two tasks for the final the evaluation.

A.6 Incremental Learning: Multi-lingual NMT

We used `transformer` architecture with our EWC regularizer implementation with the following non-default model hyper-parameters:

- embedding size: 512,
- feed-forward size: 2,048,
- attention heads: 8,
- encoder/decoder depth: 6,
- batch size: 4,096 tokens/batch,
- learning rate: 5e-4,
- label smoothing: 0.2,
- gradient clip norm: 1.0,
- warmup steps: 4,000,
- dropout 0.3,

In the follow-up multi-lingual experiments, we use Adam optimizer. The bilingual and low-resource models were trained for 200k updates. High-resource models and models trained on the full multi-lingual data were trained for 600k updates. The low-resource fine-tuning experiments were initialized by the high-resource checkpoint with the highest validation BLEU and fine-tuned for additional 200k updates. For each models, the checkpoint with the highest validation BLEU on the combined dataset from all languages was used for the final the evaluation.

A.7 Modular Transformer: String Editing

We used our implementation of `transformer_modular` architecture with the following non-default model hyper-parameters:

- embedding size: 128,
- feed-forward size: 512,
- attention heads: 8,
- encoder/decoder depth: 1,
- batch size: 4,096 tokens/batch,
- learning rate: 5e-4,
- label smoothing: 0.2,
- gradient clip norm: 1.0,
- warmup steps: 4,000,
- dropout: 0.3,
- controller initial sampling temperature: 100,
- controller final sampling temperature: 0.1,
- temperature anneal rate: 1e-9

Each model training was terminated after 100 epochs. We used exponential annealing to adjust the controller temperature during training. We used the final training checkpoint during the evaluation.

A.8 Modular Transformer: Multi-lingual NMT

We used our implementation of `transformer_modular` architecture with the following non-default model hyper-parameters:

- embedding size: 128,
- feed-forward size: 512,
- attention heads: 8,
- encoder/decoder depth: 1,

- batch size: 4,096 tokens/batch,
- learning rate: 5e-4,
- label smoothing: 0.2,
- gradient clip norm: 1.0,
- warmup steps: 4,000,
- dropout: 0.3,
- controller initial sampling temperature: 100,
- controller final sampling temperature: 0.1,
- temperature anneal rate: 1e-9

The baseline model was standard `transformer` with identical training hyper-parameters excluding the modular hyper-parameters. Each model training was terminated after 600k updates. The training checkpoint with the highest validation BLEU was used during test the evaluation.

Bibliography

Abid, A. – Farooqi, M. – Zou, J. Persistent Anti-Muslim Bias in Large Language Models. In Fourcade, M. – Kuipers, B. – Lazar, S. – Mulligan, D. K. (Ed.) *AIES '21: AAAI/ACM Conference on AI, Ethics, and Society, Virtual Event, USA, May 19-21, 2021*, p. 298–306. ACM, 2021.

Abraham, W. C. – Robins, A. Memory retention – the synaptic stability versus plasticity dilemma. *Trends in Neurosciences*. 2005, 28, 2, p. 73–78. ISSN 0166-2236.

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

Akhbardeh, F. et al. Findings of the 2021 Conference on Machine Translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, p. 1–88, Online, November 2021a. Association for Computational Linguistics.

Akhbardeh, F. et al. Findings of the 2021 Conference on Machine Translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, p. 1–88, Online, November 2021b. Association for Computational Linguistics.

Al-Jamal, D. A. H. The role of linguistic clues in medical students' reading comprehension. *Psychology Research and Behavior Management*. 2018, 11, p. 395 – 401.

Al-Rfou, R. – Choe, D. – Constant, N. – Guo, M. – Jones, L. Character-Level Language Modeling with Deeper Self-Attention. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, p. 3159–3166. AAAI Press, 2019.

Aljundi, R. – Chakravarty, P. – Tuytelaars, T. Expert Gate: Lifelong Learning with a Network of Experts. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 00, p. 7120–7129, July 2017.

Aljundi, R. – Rohrbach, M. – Tuytelaars, T. Selfless Sequential Learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Amodei, D. et al. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. In Balcan, M. F. – Weinberger, K. Q. (Ed.) *Proceedings of The 33rd International Conference on Machine Learning*, 48 / *Proceedings of Machine Learning Research*, p. 173–182, New York, New York, USA, 20–22 Jun 2016. PMLR.

Artetxe, M. – Labaka, G. – Agirre, E. – Cho, K. Unsupervised neural machine translation. In *Proceedings of the Sixth International Conference on Learning Representations*, April 2018.

Ba, L. J. – Kiros, J. R. – Hinton, G. E. Layer Normalization. *CoRR*. 2016, abs/1607.06450.

Bahdanau, D. – Cho, K. – Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*. 2014, abs/1409.0473.

Baillargeon, R. Infants' Physical World. *Current Directions in Psychological Science*. 2004, 13, 3, p. 89–94.

Bapna, A. – Firat, O. Simple, Scalable Adaptation for Neural Machine Translation. In Inui, K. – Jiang, J. – Ng, V. – Wan, X. (Ed.) *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, p. 1538–1548. Association for Computational Linguistics, 2019.

Barrault, L. et al. (Ed.). *Proceedings of the Fifth Conference on Machine Translation*, Online, November 2020. Association for Computational Linguistics.

Baziotis, C. – Titov, I. – Birch, A. – Haddow, B. Exploring Unsupervised Pretraining Objectives for Machine Translation. In Zong, C. – Xia, F. – Li, W. – Navigli, R. (Ed.) *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, ACL/IJCNLP 2021 / *Findings of ACL*, p. 2956–2971. Association for Computational Linguistics, 2021.

Belkin, M. – Hsu, D. – Ma, S. – Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*. 2019, 116, 32, p. 15849–15854.

Bellman, R. – Corporation, R. – Collection, K. M. R. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957. ISBN 9780691079516.

Beltagy, I. – Peters, M. E. – Cohan, A. Longformer: The Long-Document Transformer. *CoRR*. 2020, abs/2004.05150.

Bender, E. M. – Gebru, T. – McMillan-Major, A. – Shmitchell, S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, p. 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097.

Bengio, E. *On Reinforcement Learning for Deep Neural Architectures: Conditional Computation with Stochastic Computation Policies.* PhD thesis, McGill University, USA, 2017.

Bengio, E. – Bacon, P.-L. – Pineau, J. – Precup, D. Conditional Computation in Neural Networks for faster models, 2016.

Bengio, S. – Vinyals, O. – Jaitly, N. – Shazeer, N. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In Cortes, C. – Lawrence, N. D. – Lee, D. D. – Sugiyama, M. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, p. 1171–1179, 2015.

Bengio, Y. – Simard, P. – Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*. 1994, 5, 2, p. 157–166.

Bengio, Y. – Léonard, N. – Courville, A. C. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR*. 2013, abs/1308.3432.

Bickel, S. – Brückner, M. – Scheffer, T. Discriminative learning for differing training and test distributions. In Ghahramani, Z. (Ed.) *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, 227 / *ACM International Conference Proceeding Series*, p. 81–88. ACM, 2007.

Biesialska, M. – Biesialska, K. – Costa-jussà, M. R. Continual Lifelong Learning in Natural Language Processing: A Survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, p. 6523–6541, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

Bishop, C. *Neural Networks for Pattern Recognition.* Oxford University Press, January 1995.

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.

Bojar, O. – Tamchyna, A. CUNI in WMT15: Chimera Strikes Again. In *Proceedings of the Tenth Workshop on Statistical Machine Translation, WMT@EMNLP 2015, 17-18 September 2015, Lisbon, Portugal*, p. 79–83. The Association for Computer Linguistics, 2015.

Bojar, O. – Macháček, M. – Tamchyna, A. – Zeman, D. Scratching the Surface of Possible Translations. In Habernal, I. – Matoušek, V. (Ed.) *Text, Speech and Dialogue: 16th International Conference, TSD 2013. Proceedings*, 8082 / *Lecture Notes in Computer Science*, p. 465–474, Berlin / Heidelberg, 2013. Západočeská univerzita v Plzni, Springer Verlag. ISBN 978-3-642-40584-6.

Bojar, O. – Dušek, O. – Kocmi, T. – Libovický, J. – Novák, M. – Popel, M. – Sudarikov, R. – Variš, D. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In Sojka, P. – Horák, A. – Kopeček, I. – Pala, K. (Ed.) *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*, no. 9924 in Lecture Notes in Computer Science, p. 231–238, Cham / Heidelberg / New York / Dordrecht / London, 2016. Masaryk University, Springer International Publishing. ISBN 978-3-319-45509-9.

Bojar, O. – Kocmi, T. – Mareček, D. – Sudarikov, R. – Variš, D. CUNI Submission in WMT17: Chimera Goes Neural. In Bojar, O. (Ed.) *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, 2, p. 248–256, Stroudsburg, PA, USA, 2017. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-945626-96-8.

Boulanger-Lewandowski, N. – Bengio, Y. – Vincent, P. Audio Chord Recognition with Recurrent Neural Networks. In *ISMIR*, 2013.

Brandreth, G. *The Joy of Lex: How to Have Fun with 860,341,500 Words.* Morrow, 1980. ISBN 9780688037093.

Brown, T. et al. Language Models are Few-Shot Learners. In Larochelle, H. – Ranzato, M. – Hadsell, R. – Balcan, M. F. – Lin, H. (Ed.) *Advances in Neural Information Processing Systems*, 33, p. 1877–1901. Curran Associates, Inc., 2020a.

Brown, T. B. et al. Language Models are Few-Shot Learners. 2020b.

Bucila, C. – Caruana, R. – Niculescu-Mizil, A. Model compression. In Eliassi-Rad, T. – Ungar, L. H. – Craven, M. – Gunopulos, D. (Ed.) *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, p. 535–541. ACM, 2006.

Callison-Burch, C. – Osborne, M. – Koehn, P. Re-evaluating the Role of Bleu in Machine Translation Research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, p. 249–256, Trento, Italy, April 2006. Association for Computational Linguistics.

Carey, S. *Conceptual Change in Childhood.* MIT Press series in learning, development, and conceptual change. MIT Press, 1985.

Carey, S. The child as word learner. *Linguistic theory and psychological reality.* 1978, p. 264–293.

Caruana, R. Multitask learning. *Machine learning*. 1997, 28, 1, p. 41–75.

Cer, D. – Yang, Y. – Kong, S. – Hua, N. – Limtiaco, N. – John, R. S. – Constant, N. – Guajardo-Cespedes, M. – Yuan, S. – Tar, C. – Sung, Y. – Strope, B. – Kurzweil, R. Universal Sentence Encoder. *CoRR*. 2018, abs/1803.11175.

Chan, W. – Jaitly, N. – Le, Q. V. – Vinyals, O. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, p. 4960–4964. IEEE, 2016.

Chaudhry, A. – Dokania, P. K. – Ajanthan, T. – Torr, P. H. S. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In Ferrari, V. – Hebert, M. – Sminchisescu, C. – Weiss, Y. (Ed.) *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, 11215 / *Lecture Notes in Computer Science*, p. 556–572. Springer, 2018.

Chelba, C. – Acero, A. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Comput. Speech Lang.* 2006, 20, 4, p. 382–399.

Chen, G. – Ma, S. – Chen, Y. – Dong, L. – Zhang, D. – Pan, J. – Wang, W. – Wei, F. Zero-Shot Cross-Lingual Transfer of Neural Machine Translation with Multilingual Pretrained Encoders. In Moens, M. – Huang, X. – Specia, L. – Yih, S. W. (Ed.) *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, p. 15–26. Association for Computational Linguistics, 2021a.

Chen, G. – Ma, S. – Chen, Y. – Zhang, D. – Pan, J. – Wang, W. – Wei, F. Towards Making the Most of Cross-Lingual Transfer for Zero-Shot Neural Machine Translation. In Muresan, S. – Nakov, P. – Villavicencio, A. (Ed.) *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, p. 142–157. Association for Computational Linguistics, 2022.

Chen, S. – Zhang, Y. – Yang, Q. Multi-Task Learning in Natural Language Processing: An Overview. *CoRR*. 2021b, abs/2109.09138.

Chen, S. F. – Goodman, J. An Empirical Study of Smoothing Techniques for Language Modeling. In Joshi, A. K. – Palmer, M. (Ed.) *34th Annual Meeting of the Association for Computational Linguistics, 24-27 June 1996, University of California, Santa Cruz, California, USA, Proceedings*, p. 310–318. Morgan Kaufmann Publishers / ACL, 1996.

Chi, Z. – Huang, S. – Dong, L. – Ma, S. – Zheng, B. – Singhal, S. – Bajaj, P. – Song, X. – Mao, X. – Huang, H. – Wei, F. XLM-E: Cross-lingual Language Model Pre-training via ELECTRA. In Muresan, S. – Nakov, P. – Villavicencio, A. (Ed.) *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, p. 6170–6182. Association for Computational Linguistics, 2022.

Cho, K. – Merriënboer, B. – Gulcehre, C. – Bahdanau, D. – Bougares, F. – Schwenk, H. – Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 1724–1734, Doha, Qatar, October 2014a. Association for Computational Linguistics.

Cho, K. – Merriënboer, B. – Gulcehre, C. – Bahdanau, D. – Bougares, F. – Schwenk, H. – Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 1724–1734, Doha, Qatar, October 2014b. Association for Computational Linguistics.

Chomsky, N. Aspects of the Theory of Syntax. *Journal of Philosophy*. 1965, 64, 2, p. 67–74. doi: 10.2307/2023772.

Choshen, L. – Abend, O. Automatically Extracting Challenge Sets for Non-Local Phenomena in Neural Machine Translation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, p. 291–303, Hong Kong, China, November 2019. Association for Computational Linguistics.

Chu, C. – Dabre, R. – Kurohashi, S. An Empirical Comparison of Domain Adaptation Methods for Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, p. 385–391, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Cichon, J. – Gan, W.-B. Branch-specific dendritic Ca2+ spikes cause persistent synaptic plasticity. *Nature*. 03 2015, 520.

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

Conneau, A. – Lample, G. Cross-lingual Language Model Pretraining. In Wallach, H. M. – Larochelle, H. – Beygelzimer, A. – d'Alché-Buc, F. – Fox, E. B. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, p. 7057–7067, 2019a.

Conneau, A. – Lample, G. Cross-lingual Language Model Pretraining. In Wallach, H. M. – Larochelle, H. – Beygelzimer, A. – d'Alché-Buc, F. – Fox, E. B. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, p. 7057–7067, 2019b.

Correia, G. M. – Niculae, V. – Martins, A. F. T. Adaptively Sparse Transformers. 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. 2174–2184, Hong Kong, China, November 2019. Association for Computational Linguistics.

Csordás, R. – Steenkiste, S. – Schmidhuber, J. Are Neural Nets Modular? Inspecting Functional Modularity Through Differentiable Weight Masks. In *International Conference on Learning Representations*, 2021.

Csordás, R. – Steenkiste, S. – Schmidhuber, J. Are Neural Nets Modular? Inspecting Functional Modularity Through Differentiable Weight Masks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.* 1989, 2, 4, p. 303–314.

Dai, W. – Yang, Q. – Xue, G. – Yu, Y. Boosting for transfer learning. In Ghahramani, Z. (Ed.) *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, 227 / *ACM International Conference Proceeding Series*, p. 193–200. ACM, 2007.

Dai, Z. – Yang, Z. – Yang, Y. – Carbonell, J. G. – Le, Q. V. – Salakhutdinov, R. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In Korhonen, A. – Traum, D. R. – Màrquez, L. (Ed.) *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, p. 2978–2988. Association for Computational Linguistics, 2019.

Daumé III, H. Bayesian Multitask Learning with Latent Hierarchies. In Bilmes, J. A. – Ng, A. Y. (Ed.) *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, p. 135–142. AUAI Press, 2009.

Masson d'Autume, C. – Ruder, S. – Kong, L. – Yogatama, D. Episodic Memory in Lifelong Language Learning. In Wallach, H. M. – Larochelle, H. – Beygelzimer, A. – d'Alché-Buc, F. – Fox, E. B. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, p. 13122–13131, 2019a.

Masson d'Autume, C. – Ruder, S. – Kong, L. – Yogatama, D. Episodic Memory in Lifelong Language Learning. In Wallach, H. M. – Larochelle, H. – Beygelzimer, A. – d'Alché-Buc, F. – Fox, E. B. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, p. 13122–13131, 2019b.

Dehghani, M. – Gouws, S. – Vinyals, O. – Uszkoreit, J. – Kaiser, L. Universal Transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*. 2012, 29, 6, p. 141–142.

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

Dong, D. – Wu, H. – He, W. – Yu, D. – Wang, H. Multi-Task Learning for Multiple Language 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. 1723–1732, Beijing, China, July 2015. Association for Computational Linguistics.

Draelos, T. J. – Miner, N. E. – Lamb, C. C. – Cox, J. A. – Vineyard, C. M. – Carlson, K. D. – Severa, W. M. – James, C. D. – Aimone, J. B. Neurogenesis deep learning: Extending deep networks to accommodate new classes. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, p. 526–533. IEEE, 2017.

Dredze, M. – Crammer, K. Online Methods for Multi-Domain Learning and Adaptation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, p. 689–697, Honolulu, Hawaii, October 2008. Association for Computational Linguistics.

Eigen, D. – Ranzato, M. – Sutskever, I. Learning Factored Representations in a Deep Mixture of Experts. In Bengio, Y. – LeCun, Y. (Ed.) *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, 2014.

Elman, J. L. Finding Structure in Time. *Cognitive Science*. 1990, 14, 2, p. 179–211.

Escolano, C. – Costa-jussà, M. R. – Fonollosa, J. A. R. – Artetxe, M. Multilingual Machine Translation: Closing the Gap between Shared and Language-specific Encoder-Decoders. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, p. 944–948, Online, April 2021. Association for Computational Linguistics.

Feynman, R. P. Space-Time Approach to Non-Relativistic Quantum Mechanics. *Rev. Mod. Phys.* Apr 1948, 20, p. 367–387.

Firat, O. – Sankaran, B. – Al-onaizan, Y. – Yarman Vural, F. T. – Cho, K. Zero-Resource Translation with Multi-Lingual Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, p. 268–277, Austin, Texas, November 2016. Association for Computational Linguistics.

Fodor, J. A. – Pylyshyn, Z. W. Connectionism and cognitive architecture: A critical analysis. *Cognition*. 1988, 28, p. 3–71.

Freitag, M. – Al-Onaizan, Y. Fast Domain Adaptation for Neural Machine Translation. *CoRR*. 2016, abs/1612.06897.

Freitag, M. – Al-Onaizan, Y. Beam Search Strategies for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, p. 56–60, Vancouver, August 2017. Association for Computational Linguistics.

French, R. M. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*. 1999, 3, p. 128–135.

Fung, I. – Mak, B. Multi-Head Attention for End-to-End Neural Machine Translation. In *11th International Symposium on Chinese Spoken Language Processing, ISCSLP 2018, Taipei City, Taiwan, November 26-29, 2018*, p. 250–254. IEEE, 2018.

Gal, Y. – Ghahramani, Z. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In Lee, D. D. – Sugiyama, M. – Luxburg, U. – Guyon, I. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, p. 1019–1027, 2016.

Galley, M. – Manning, C. D. A Simple and Effective Hierarchical Phrase Reordering Model. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, p. 848–856. ACL, 2008.

Gehman, S. – Gururangan, S. – Sap, M. – Choi, Y. – Smith, N. A. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. In *EMNLP (Findings)*, p. 3356–3369. Association for Computational Linguistics, 2020.

Gehring, J. – Auli, M. – Grangier, D. – Yarats, D. – Dauphin, Y. N. Convolutional Sequence to Sequence Learning. In Precup, D. – Teh, Y. W. (Ed.) *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 70 / *Proceedings of Machine Learning Research*, p. 1243–1252. PMLR, 2017.

Gelman, A. – Lee, D. – Guo, J. Stan: A Probabilistic Programming Language for Bayesian Inference and Optimization. *Journal of Educational and Behavioral Statistics*. 2015, 40, 5, p. 530–543.

Geva, M. – Malmi, E. – Szpektor, I. – Berant, J. DiscoFuse: A Large-Scale Dataset for Discourse-Based Sentence Fusion. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, p. 3443–3455, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Ghahramani, Z. Probabilistic machine learning and artificial intelligence. *Nature*. May 2015, 521, 7553, p. 452–459.

Goodfellow, I. – Bengio, Y. – Courville, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Goodman, N. D. – Mansinghka, V. K. – Roy, D. M. – Bonawitz, K. A. – Tenenbaum, J. B. Church: a language for generative models. In McAllester, D. A. – Myllymäki, P. (Ed.) *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*, p. 220–229. AUAI Press, 2008.

Gopnik, A. – Meltzoff, A. *Words, Thoughts, and Theories*. A Bradford book. MIT Press, 1997. ISBN 9780262071758.

Gorman, K. – Bedrick, S. We Need to Talk about Standard Splits. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 2786–2791, Florence, Italy, July 2019. Association for Computational Linguistics.

Goyal, N. – Du, J. – Ott, M. – Anantharaman, G. – Conneau, A. Larger-Scale Transformers for Multilingual Masked Language Modeling. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, p. 29–33, Online, August 2021. Association for Computational Linguistics.

Graves, A. Sequence Transduction with Recurrent Neural Networks. *CoRR*. 2012, abs/1211.3711.

Grießhaber, D. – Maucher, J. – Vu, N. T. Fine-tuning BERT for Low-Resource Natural Language Understanding via Active Learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, p. 1158–1171, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

Grundkiewicz, R. – Heafield, K. Neural Machine Translation Techniques for Named Entity Transliteration. In *Proceedings of the Seventh Named Entities Workshop*, p. 89–94, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Gū, J. – Shavarani, H. S. – Sarkar, A. Top-down Tree Structured Decoding with Syntactic Connections for Neural Machine Translation and Parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 401–413, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

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.

Gull, S. F. *Bayesian Inductive Inference and Maximum Entropy*, p. 53–74. Springer Netherlands, Dordrecht, 1988. ISBN 978-94-009-3049-0.

Guo, H. – Pasunuru, R. – Bansal, M. Dynamic Multi-Level Multi-Task Learning for Sentence Simplification. In Bender, E. M. – Derczynski, L. – Isabelle, P. (Ed.) *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, p. 462–476. Association for Computational Linguistics, 2018.

Gururangan, S. – Marasovic, A. – Swayamdipta, S. – Lo, K. – Beltagy, I. – Downey, D. – Smith, N. A. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In Jurafsky, D. – Chai, J. – Schluter, N. – Tetreault, J. R. (Ed.) *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, p. 8342–8360. Association for Computational Linguistics, 2020.

Ha, T.-L. – Niehues, J. – Waibel, A. Toward Multilingual Neural Machine Translation with Universal Encoder and Decoder. In *Proceedings of the 13th International Conference on Spoken Language Translation*, Seattle, Washington D.C, December 8-9 2016. International Workshop on Spoken Language Translation.

Harris, Z. S. Distributional Structure. *WORD*. 1954, 10, 2-3, p. 146–162.

Hashimoto, K. – Xiong, C. – Tsuruoka, Y. – Socher, R. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, p. 1923–1933, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Hassibi, B. – Stork, D. G. Second Order Derivatives for Network Pruning: Optimal Brain Surgeon. In Hanson, S. J. – Cowan, J. D. – Giles, C. L. (Ed.) *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*, p. 164–171. Morgan Kaufmann, 1992.

Hastie, T. – Tibshirani, R. – Friedman, J. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., 2001.

He, K. – Zhang, X. – Ren, S. – Sun, J. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, p. 770–778. IEEE Computer Society, 2016a.

He, K. – Zhang, X. – Ren, S. – Sun, J. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, p. 770–778. IEEE Computer Society, 2016b.

Heafield, K. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, p. 187–197, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.

Hecht-Nielsen, R. *Theory of the Backpropagation Neural Network*, p. 65–93. Harcourt Brace & Co., USA, 1992. ISBN 0127412522.

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

Helcl, J. – Libovický, J. – Kocmi, T. – Musil, T. – Cífka, O. – Variš, D. – Bojar, O. Neural Monkey: The Current State and Beyond. In Neubig, G. – Cherry, C. (Ed.) *The 13th Conference of The Association for Machine Translation in the Americas, Vol. 1: MT Researchers' Track*, p. 168–176, Stroudsburg, PA, USA, 2018a. The Association for Machine Translation in the Americas, The Association for Machine Translation in the Americas.

Helcl, J. – Libovický, J. – Variš, D. CUNI System for the WMT18 Multimodal Translation Task. In Bojar, O. (Ed.) *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Tasks*, 2, p. 622–629, Stroudsburg, PA, USA, 2018b. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-948087-81-0.

Hiebert, E. – Kamil, M. *Teaching and Learning Vocabulary: Bringing Research to Practice*. L. Erlbaum Associates, 2005. ISBN 9780805852851.

Hinton, G. E. – Plaut, D. C. Using Fast Weights to Deblur Old Memories. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, p. 177–186, Hillsdale, NJ, 1987. Erlbaum.

Hinton, G. E. – McClelland, J. L. – Rumelhart, D. E. Distributed Representations. In Boden, M. A. (Ed.) *The Philosophy of Artificial Intelligence*, Oxford readings in philosophy. Oxford, UK: Oxford University Press, 1990. p. 248–280.

Hinton, G. E. – Vinyals, O. – Dean, J. Distilling the Knowledge in a Neural Network. *CoRR*. 2015, abs/1503.02531.

Hochreiter, S. – Bengio, Y. – Frasconi, P. – Schmidhuber, J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer, S. C. – Kolen, J. F. (Ed.) *A Field Guide to Dynamical Recurrent Neural Networks*. Piscataway, NJ, USA: IEEE Press, 2001.

Hochreiter, S. The Vanishing Gradient Problem during Learning Recurrent Neural Nets and Problem Solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* April 1998, 6, 2, p. 107–116. ISSN 0218-4885.

Hochreiter, S. – Schmidhuber, J. Long Short-Term Memory. *Neural Computation*. 11 1997, 9, 8, p. 1735–1780. ISSN 0899-7667.

Hoefler, T. – Alistarh, D. – Ben-Nun, T. – Dryden, N. – Peste, A. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*. 09 2021, 22, 241, p. 1–124.

Hoffer, E. – Hubara, I. – Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In Guyon, I. – Luxburg, U. V. – Bengio, S. – Wallach, H. – Fergus, R. – Vishwanathan, S. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems*, 30. Curran Associates, Inc., 2017.

Hornik, K. – Stinchcombe, M. – White, H. Multilayer Feedforward Networks Are Universal Approximators. *Neural Netw.* jul 1989, 2, 5, p. 359–366. ISSN 0893-6080.

Hubel, D. H. – Wiesel, T. N. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*. 1959, 148.

Huszár, F. On Quadratic Penalties in Elastic Weight Consolidation. *CoRR*. 2017, abs/1712.03847.

Ivakhnenko, A. – Lapa, V. *Cybernetic Predicting Devices*. Jprs report. CCM Information Corporation, 1973.

Iyyer, M. – Manjunatha, V. – Boyd-Graber, J. – Daumé III, H. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. 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. 1681–1691, Beijing, China, July 2015. Association for Computational Linguistics.

Jacobs, R. A. – Jordan, M. I. – Nowlan, S. J. – Hinton, G. E. Adaptive Mixtures of Local Experts. *Neural Comput.* March 1991, 3, 1, p. 79–87. ISSN 0899-7667.

Jastrzebski, S. – Kenton, Z. – Arpit, D. – Ballas, N. – Fischer, A. – Bengio, Y. – Storkey, A. J. Three Factors Influencing Minima in SGD. *CoRR*. 2017, abs/1711.04623.

Jean, S. – Firat, O. – Cho, K. – Memisevic, R. – Bengio, Y. Montreal Neural Machine Translation Systems for WMT'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, p. 134–140, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Jern, A. – Kemp, C. A probabilistic account of exemplar and category generation. *Cognitive Psychology*. 2013, 66, 1, p. 85–125. ISSN 0010-0285.

Jia, C. – Yang, Y. – Xia, Y. – Chen, Y. – Parekh, Z. – Pham, H. – Le, Q. V. – Sung, Y. – Li, Z. – Duerig, T. Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision. In Meila, M. – Zhang, T. (Ed.) *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, 139 / *Proceedings of Machine Learning Research*, p. 4904–4916. PMLR, 2021.

Johnson, M. – Schuster, M. – Le, Q. V. – Krikun, M. – Wu, Y. – Chen, Z. – Thorat, N. – Viégas, F. B. – Wattenberg, M. – Corrado, G. – Hughes, M. – Dean, J. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Trans. Assoc. Comput. Linguistics*. 2017, 5, p. 339–351.

Joshi, M. – Choi, E. – Weld, D. – Zettlemoyer, L. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Karpathy, A. – Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, p. 3128–3137. IEEE Computer Society, 2015.

Katharopoulos, A. – Vyas, A. – Pappas, N. – Fleuret, F. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In III, H. D. – Singh, A. (Ed.) *Proceedings of the 37th International Conference on Machine Learning*, 119 / *Proceedings of Machine Learning Research*, p. 5156–5165. PMLR, 13–18 Jul 2020.

Kemker, R. – McClure, M. – Abitino, A. – Hayes, T. L. – Kanan, C. Measuring Catastrophic Forgetting in Neural Networks. In McIlraith, S. A. – Weinberger, K. Q. (Ed.) *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, p. 3390–3398. AAAI Press, 2018.

Khayrallah, H. – Thompson, B. – Duh, K. – Koehn, P. Regularized Training Objective for Continued Training for Domain Adaptation in Neural Machine Translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, p. 36–44, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Kim, D. – Saito, K. – Saenko, K. – Sclaroff, S. – Plummer, B. A. MULE: Multimodal Universal Language Embedding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, p. 11254–11261. AAAI Press, 2020.

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.

Kingma, D. P. – Ba, J. Adam: A Method for Stochastic Optimization. *CoRR*. 2014, abs/1412.6980.

Kingma, D. P. – Welling, M. Auto-Encoding Variational Bayes. In Bengio, Y. – LeCun, Y. (Ed.) *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Kirkpatrick, J. – Pascanu, R. – Rabinowitz, N. C. – Veness, J. – Desjardins, G. – Rusu, A. A. – Milan, K. – Quan, J. – Ramalho, T. – Grabska-Barwinska, A. – Hassabis, D. – Clopath, C. – Kumaran, D. – Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*. 2017, 114 13, p. 3521–3526.

Kirsch, L. – Kunze, J. – Barber, D. Modular Networks: Learning to Decompose Neural Computation. In Bengio, S. – Wallach, H. – Larochelle, H. – Grauman, K. – Cesa-Bianchi, N. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems 31*, p. 2408–2418. Curran Associates, Inc., 2018.

Kocmi, T. – Bojar, O. Trivial Transfer Learning for Low-Resource Neural Machine Translation. In Bojar, O. et al. (Ed.) *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, p. 244–252. Association for Computational Linguistics, 2018.

Kocmi, T. – Bojar, O. Efficiently Reusing Old Models Across Languages via Transfer Learning. In Forcada, M. L. – Martins, A. – Moniz, H. – Turchi, M. – Bisazza, A. – Moorkens, J. – Arenas, A. G. – Nurminen, M. – Marg, L. – Fumega, S. – Martins, B. – Batista, F. – Coheur, L. – Escartín, C. P. – Trancoso, I. (Ed.) *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, EAMT 2020, Lisboa, Portugal, November 3-5, 2020*, p. 19–28. European Association for Machine Translation, 2020.

Kocmi, T. – Variš, D. – Bojar, O. CUNI Basque-to-English Submission in IWSLT18. In Turchi, M. – Niehues, J. – Federico, M. (Ed.) *Proceedings of the International Workshop on Spoken Language Translation*, p. 142–146, Karlsruhe, Germany, 2018. University of Brugge, Karlsruhe Institute of Technology.

Kocmi, T. – Popel, M. – Bojar, O. Announcing CzEng 2.0 Parallel Corpus with over 2 Gigawords. *arXiv preprint arXiv:2007.03006*. 2020.

Koehn, P. – Knowles, R. Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, p. 28–39, Vancouver, August 2017. Association for Computational Linguistics.

Koehn, P. – Axelrod, A. – Birch, A. – Callison-Burch, C. – Osborne, M. – Talbot, D. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *2005 International Workshop on Spoken Language Translation, IWSLT 2005, Pittsburgh, PA, USA, October 24-25, 2005*, p. 68–75. ISCA, 2005.

Kolen, J. F. – Kremer, S. C. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, p. 237–243. 2001.

Kondo, S. – Hotate, K. – Hirasawa, T. – Kaneko, M. – Komachi, M. Sentence Concatenation Approach to Data Augmentation for Neural Machine Translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, p. 143–149, Online, June 2021. Association for Computational Linguistics.

Krizhevsky, A. – Hinton, G. – others. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A. – Sutskever, I. – Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In Bartlett, P. L. – Pereira, F. C. N. – Burges, C. J. C. – Bottou, L. – Weinberger, K. Q. (Ed.) *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, p. 1106–1114, 2012.

Kudo, T. – Richardson, J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, p. 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics.

Kutalev, A. – Lapina, A. Stabilizing Elastic Weight Consolidation method in practical ML tasks and using weight importances for neural network pruning. *CoRR*. 2021, abs/2109.10021.

Lachter, J. – Bever, T. The relation between linguistic structure and associative theories of language learning-A constructive critique of some connectionist learning models. *Cognition*. March 1988, 28, 1-2, p. 195–247. ISSN 0010-0277.

Lake, B. M. – Ullman, T. D. – Tenenbaum, J. B. – Gershman, S. J. Building machines that learn and think like people. *Behavioral and Brain Sciences*. 2017, 40, p. e253. doi: 10.1017/S0140525X16001837.

Lample, G. – Denoyer, L. – Ranzato, M. Unsupervised Machine Translation Using Monolingual Corpora Only. *CoRR*. 2017, abs/1711.00043.

Landau, B. – Smith, L. B. – Jones, S. S. The importance of shape in early lexical learning. *Cognitive Development*. 1988, 3, 3, p. 299–321. ISSN 0885-2014.

Lecun, Y. – Bengio, Y. *Convolutional networks for images, speech, and time-series*. MIT Press, 1995.

LeCun, Y. – Denker, J. S. – Solla, S. A. Optimal Brain Damage. In Touretzky, D. S. (Ed.) *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, p. 598–605. Morgan Kaufmann, 1989.

Lee, J. – Cho, K. – Hofmann, T. Fully Character-Level Neural Machine Translation without Explicit Segmentation. *Transactions of the Association for Computational Linguistics*. 2017, 5, p. 365–378.

Li, Z. – Hoiem, D. Learning Without Forgetting. In *European Conference on Computer Vision*, p. 614–629. Springer, 2016.

Li, Z. – Wallace, E. – Shen, S. – Lin, K. – Keutzer, K. – Klein, D. – Gonzalez, J. Train Big, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers. In III, H. D. – Singh, A. (Ed.) *Proceedings of the 37th International Conference on Machine Learning*, 119 / *Proceedings of Machine Learning Research*, p. 5958–5968. PMLR, 13–18 Jul 2020.

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.

Lin, C.-Y. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, p. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Lin, C.-Y. – Och, F. J. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, p. 605–612, Barcelona, Spain, July 2004.

Lin, T.-Y. – Maire, M. – Belongie, S. – Hays, J. – Perona, P. – Ramanan, D. – Dollár, P. – Zitnick, C. L. Microsoft COCO: Common Objects in Context. In Fleet, D. – Pajdla, T. – Schiele, B. – Tuytelaars, T. (Ed.) *Computer Vision – ECCV 2014*, p. 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

Lin, T. – Goyal, P. – Girshick, R. B. – He, K. – Dollár, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 2020, 42, 2, p. 318–327.

Liu, Y. – Ott, M. – Goyal, N. – Du, J. – Joshi, M. – Chen, D. – Levy, O. – Lewis, M. – Zettlemoyer, L. – Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*. 2019, abs/1907.11692.

Liu, Y. – Gu, J. – Goyal, N. – Li, X. – Edunov, S. – Ghazvininejad, M. – Lewis, M. – Zettlemoyer, L. Multilingual Denoising Pre-training for Neural Machine Translation. *Trans. Assoc. Comput. Linguistics*. 2020, 8, p. 726–742.

Lloyd, J. – Duvenaud, D. – Grosse, R. – Tenenbaum, J. – Ghahramani, Z. Automatic Construction and Natural-Language Description of Nonparametric Regression Models. *Proceedings of the AAAI Conference on Artificial Intelligence*. Jun. 2014, 28, 1.

Lu, J. – Xiong, C. – Parikh, D. – Socher, R. Knowing When to Look: Adaptive Attention via a Visual Sentinel for Image Captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, p. 3242–3250. IEEE Computer Society, 2017.

Lu, Y. – Keung, P. – Ladhak, F. – Bhardwaj, V. – Zhang, S. – Sun, J. A neural interlingua for multilingual machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, p. 84–92, Brussels, Belgium, October 2018. Association for Computational Linguistics.

Luong, M. – Manning, C. D. Stanford neural machine translation systems for spoken language domains. In Federico, M. – Stüker, S. – Niehues, J. (Ed.) *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign@IWSLT 2015, Da Nang, Vietnam, December 3-4, 2015*, 2015.

Luong, M. – Le, Q. V. – Sutskever, I. – Vinyals, O. – Kaiser, L. Multi-task Sequence to Sequence Learning. In Bengio, Y. – LeCun, Y. (Ed.) *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Luong, T. – Pham, H. – Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, p. 1412–1421, Lisbon, Portugal, September 2015a. Association for Computational Linguistics.

Luong, T. – Pham, H. – Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, p. 1412–1421, Lisbon, Portugal, September 2015b. Association for Computational Linguistics.

MacKay, D. J. C. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*. May 1992, 4, 3, p. 448–472. ISSN 0899-7667.

Maddison, C. J. – Mnih, A. – Teh, Y. W. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Madhyastha, P. S. – Wang, J. – Specia, L. End-to-end Image Captioning Exploits Distributional Similarity in Multimodal Space. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

Mareček, D. – Rosa, R. From Balustrades to Pierre Vinken: Looking for Syntax in Transformer Self-Attentions. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, p. 263–275, Florence, Italy, August 2019. Association for Computational Linguistics.

Markman, E. M. *Categorization and Naming in Children: Problems of Induction*. The MIT Press, 02 1991. ISBN 9780262279147.

Martin, L. – Müller, B. – Suárez, P. J. O. – Dupont, Y. – Romary, L. – Clergerie, É. – Seddah, D. – Sagot, B. CamemBERT: a Tasty French Language Model. In Jurafsky, D. – Chai, J. – Schluter, N. – Tetreault, J. R. (Ed.) *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, p. 7203–7219. Association for Computational Linguistics, 2020.

McCann, B. – Keskar, N. S. – Xiong, C. – Socher, R. The Natural Language Decathlon: Multitask Learning as Question Answering. *CoRR*. 2018, abs/1806.08730.

McClelland, J. L. – Rumelhart, D. E. – Hinton, G. E. *The Appeal of Parallel Distributed Processing*, p. 3–44. MIT Press, Cambridge, MA, USA, 1986. ISBN 026268053X.

McCloskey, M. – Cohen, N. J. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *The Psychology of Learning and Motivation*. 1989, 24, p. 104–169.

McCullagh, P. *Generalized Linear Models*. CRC Press, 2019. ISBN 9781351445849.

Mhaskar, S. – Jain, A. – Banerjee, A. – Bhattacharyya, P. Multilingual Machine Translation Systems at WAT 2021: One-to-Many and Many-to-One Transformer based NMT. In *Proceedings of the 8th Workshop on Asian Translation (WAT2021)*, p. 233–237, Online, August 2021. Association for Computational Linguistics.

Miceli Barone, A. V. – Haddow, B. – Germann, U. – Sennrich, R. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, p. 1489–1494, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Michel, P. – Levy, O. – Neubig, G. Are Sixteen Heads Really Better than One? In Wallach, H. – Larochelle, H. – Beygelzimer, A. – Alché-Buc, F. – Fox, E. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems*, 32, p. 14014–14024. Curran Associates, Inc., 2019.

Mikolov, T. – Chen, K. – Corrado, G. – Dean, J. Efficient Estimation of Word Representations in Vector Space. In Bengio, Y. – LeCun, Y. (Ed.) *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

Mikolov, T. – Karafiát, M. – Burget, L. – Černocký, J. – Khudanpur, S. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, 2010, p. 1045–1048. International Speech Communication Association, 2010. ISBN 978-1-61782-123-3.

Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature*. February 2015, 518, 7540, p. 529–533. ISSN 00280836.

Murphy, G. L. Comprehending Complex Concepts. *Cogn. Sci.* 1988, 12, 4, p. 529–562.

Murphy, G. – Medin, D. The Role of Theories in Conceptual Coherence. *Psychological Review*. July 1985, 92, 3, p. 289–316. ISSN 0033-295X.

Murphy, G. – Ross, B. Predictions from uncertain categorizations. *Cognitive Psychology*. October 1994, 27, 2, p. 148–193. ISSN 0010-0285.

Nair, V. – Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. ICML'10, p. 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Neal, R. M. – Hinton, G. E. A View of the Em Algorithm that Justifies Incremental, Sparse, and other Variants. In Jordan, M. I. (Ed.) *Learning in Graphical Models*, 89 / *NATO ASI Series*. Cambridge, MA, USA: Springer Netherlands, 1998. p. 355–368. ISBN 978-94-010-6104-9.

Neishi, M. – Yoshinaga, N. On the Relation between Position Information and Sentence Length in Neural Machine Translation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, p. 328–338, Hong Kong, China, November 2019. Association for Computational Linguistics.

Ni, M. – Huang, H. – Su, L. – Cui, E. – Bharti, T. – Wang, L. – Zhang, D. – Duan, N. M3P: Learning Universal Representations via Multitask Multilingual Multimodal Pre-Training. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, p. 3977–3986. Computer Vision Foundation / IEEE, 2021.

Osherson, D. N. – Smith, E. E. On the adequacy of prototype theory as a theory of concepts. *Cognition*. 1981, 9, 1, p. 35–58. ISSN 0010-0277.

Ott, M. – Edunov, S. – Baevski, A. – Fan, A. – Gross, S. – Ng, N. – Grangier, D. – Auli, M. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Pan, S. J. – Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* 2010, 22, 10, p. 1345–1359.

Panevová, J. On Verbal Frames in Functional Generative Description, Part I. *The Prague Bulletin of Mathematical Linguistics*. 1974, 22, p. 3–40.

Panevová, J. Valency Frames and the Meaning of the Sentence. In Luelsdorff, P. A. (Ed.) *The Prague School of Structural and Functional Linguistics*, p. 223–243, Amsterdam-Philadelphia, 1994. John Benjamins Publishing Company.

Papineni, K. – Roukos, S. – Ward, T. – Zhu, W.-J. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, p. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

Parisi, G. I. – Kemker, R. – Part, J. L. – Kanan, C. – Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*. 2019, 113, p. 54–71.

Pascanu, R. – Mikolov, T. – Bengio, Y. On the difficulty of training recurrent neural networks. In Dasgupta, S. – McAllester, D. (Ed.) *Proceedings of the 30th International Conference on Machine Learning*, 28 / *Proceedings of Machine Learning Research*, p. 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Pasunuru, R. – Bansal, M. Continual and Multi-Task Architecture Search. In Korhonen, A. – Traum, D. R. – Màrquez, L. (Ed.) *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, p. 1911–1922. Association for Computational Linguistics, 2019. ISBN 978-1-950737-48-2.

Peters, M. E. – Neumann, M. – Iyyer, M. – Gardner, M. – Clark, C. – Lee, K. – Zettlemoyer, L. Deep Contextualized Word Representations. In Walker, M. A. – Ji, H. – Stent, A. (Ed.) *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, p. 2227–2237. Association for Computational Linguistics, 2018.

Peters, M. E. – Ruder, S. – Smith, N. A. To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, p. 7–14, Florence, Italy, August 2019. Association for Computational Linguistics.

Pfeiffer, J. – Vulic, I. – Gurevych, I. – Ruder, S. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In Webber, B. – Cohn, T. – He, Y. – Liu, Y. (Ed.) *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, p. 7654–7673. Association for Computational Linguistics, 2020.

Pinker, S. – Prince, A. On Language and Connectionism: Analysis of a Parallel Distributed Processing Model of Language Acquisition. *COGNITION*. 1988, 28, p. 73–193.

Polikar, R. – Upda, L. – Upda, S. S. – Honavar, V. G. Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Trans. Syst. Man Cybern. Part C*. 2001, 31, 4, p. 497–508.

Popel, M. – Tomkova, M. – Tomek, J. – Kaiser – Uszkoreit, J. – Bojar, O. – Žabokrtský, Z. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nature Communications*. 2020, 11, 4381, p. 1–15. ISSN 2041-1723.

Post, M. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, p. 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics.

Press, O. – Wolf, L. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, p. 157–163, Valencia, Spain, April 2017. Association for Computational Linguistics.

Rajpurkar, P. – Zhang, J. – Lopyrev, K. – Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, p. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

Ramachandran, P. – Liu, P. J. – Le, Q. V. Unsupervised Pretraining for Sequence to Sequence Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, p. 383–391, 2017.

Ramezani, M. – Marble, K. – Trang, H. – Johnsrude, I. – Abolmaesumi, P. Joint Sparse Representation of Brain Activity Patterns in Multi-Task fMRI Data. *IEEE transactions on medical imaging*. 07 2014, 34.

Ranzato, M. – Chopra, S. – Auli, M. – Zaremba, W. Sequence Level Training with Recurrent Neural Networks. In Bengio, Y. – LeCun, Y. (Ed.) *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Rebuffi, S.-A. – Kolesnikov, A. I. – Sperl, G. – Lampert, C. H. iCaRL: Incremental Classifier and Representation Learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, p. 5533–5542.

Redmon, J. – Divvala, S. K. – Girshick, R. B. – Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, p. 779–788. IEEE Computer Society, 2016.

Rei, R. – Stewart, C. – Farinha, A. C. – Lavie, A. COMET: A Neural Framework for MT Evaluation. In Webber, B. – Cohn, T. – He, Y. – Liu, Y. (Ed.) *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, p. 2685–2702. Association for Computational Linguistics, 2020.

Ren, S. – He, K. – Girshick, R. – Sun, J. Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, p. 91–99, Cambridge, MA, USA, 2015. MIT Press.

Ring, M. B. *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin, TX, USA, 1995.

Ripley, B. D. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996. doi: 10.1017/CBO9780511812651.

Rips, L. – Hespos, S. Divisions of the physical world: Concepts of objects and substances. *Psychological Bulletin*. July 2015, 141, 4, p. 786–811. ISSN 0033-2909.

Rips, L. Inductive judgments about natural categories. *Journal of Memory and Language*. December 1975, 14, 6, p. 665–681. ISSN 0749-596X.

Robbins, H. – Monro, S. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*. 1951, 22, 3, p. 400 – 407. Available at: https://doi.org/10.1214/aoms/1177729586.

Robins, A. V. Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connect. Sci.* 1995, 7, 2, p. 123–146.

Rosenbaum, C. – Klinger, T. – Riemer, M. Routing Networks: Adaptive Selection of Non-Linear Functions for Multi-Task Learning. In *International Conference on Learning Representations*, 2018.

Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*. 1958, 65, 6, p. 386–408. ISSN 0033-295X.

Rothe, S. – Narayan, S. – Severyn, A. Leveraging Pre-trained Checkpoints for Sequence Generation Tasks. *Transactions of the Association for Computational Linguistics*. 2020, 8, p. 264–280.

Rumelhart, D. E. – McClelland, J. L. *Learning Internal Representations by Error Propagation*, p. 318–362. 1987.

Rumelhart, D. – Hinton, G. – Williams, R. Learning representations by back-propagating errors. *Nature*. 1986, 323, 6088, p. 533–536.

Scherer, D. – Müller, A. – Behnke, S. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In Diamantaras, K. – Duch, W. – Iliadis, L. S. (Ed.) *Artificial Neural Networks – ICANN 2010*, p. 92–101, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

Schlimmer, J. C. – Granger, R. H. Incremental Learning from Noisy Data. *Mach. Learn.* 1986, 1, 3, p. 317–354.

Schraudolph, N. N. Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent. *Neural Comput.* 2002a, 14, 7, p. 1723–1738.

Schraudolph, N. N. Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent. *Neural Comput.* 2002b, 14, 7, p. 1723–1738.

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 2016a. Association for Computational Linguistics.

Sennrich, R. – Haddow, B. – Birch, A. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 1715–1725, Berlin, Germany, August 2016b. Association for Computational Linguistics.

Sewak, M. *Deep Q Network (DQN), Double DQN, and Dueling DQN*, p. 95–108. Springer Singapore, Singapore, 2019. ISBN 978-981-13-8285-7.

Shannon, C. E. A mathematical theory of communication. *Bell Syst. Tech. J.* 1948, 27, 3, p. 379–423.

Shaw, P. – Uszkoreit, J. – Vaswani, A. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, p. 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

Shazeer, N. – Stern, M. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In Dy, J. G. – Krause, A. (Ed.) *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 80 / *Proceedings of Machine Learning Research*, p. 4603–4611. PMLR, 2018.

Shazeer, N. – Mirhoseini, A. – Maziarz, K. – Davis, A. – Le, Q. V. – Hinton, G. E. – Dean, J. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Shen, S. – Cheng, Y. – He, Z. – He, W. – Wu, H. – Sun, M. – Liu, Y. Minimum Risk Training for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

Shi, X. – Knight, K. – Yuret, D. Why Neural Translations are the Right Length. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, p. 2278–2282, Austin, Texas, November 2016. Association for Computational Linguistics.

Shin, H. – Lee, J. K. – Kim, J. – Kim, J. Continual Learning with Deep Generative Replay. In Guyon, I. – Luxburg, U. – Bengio, S. – Wallach, H. M. – Fergus, R. – Vishwanathan, S. V. N. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, p. 2990–2999, 2017.

Silver, D. L. – Mercer, R. E. The Task Rehearsal Method of Life-Long Learning: Overcoming Impoverished Data. In Cohen, R. – Spencer, B. (Ed.) *Advances in Artificial Intelligence, 15th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2002, Calgary, Canada, May 27-29, 2002, Proceedings*, 2338 / *Lecture Notes in Computer Science*, p. 90–101. Springer, 2002.

Silver, D. L. – Yang, Q. – Li, L. Lifelong Machine Learning Systems: Beyond Learning Algorithms. In *Lifelong Machine Learning, Papers from the 2013 AAAI Spring Symposium, Palo Alto, California, USA, March 25-27, 2013*, SS-13-05 / *AAAI Technical Report*. AAAI, 2013.

Sodhani, S. – Chandar, S. – Bengio, Y. Toward Training Recurrent Neural Networks for Lifelong Learning. *Neural Comput.* 2020, 32, 1, p. 1–35.

Søgaard, A. – Ebert, S. – Bastings, J. – Filippova, K. We Need to Talk About Random Splits. *CoRR*. 2020, abs/2005.00636.

Solomonoff, R. J. A system for machine learning based on algorithmic probability. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, November 14-17, 1989, Cambridge, Massachusetts, USA*, p. 298–299. IEEE, 1989.

Spelke, E. S. Principles of Object Perception. *Cognitive Science*. 1990, p. 29–56.

Sporns – Betzel. Modular Brain Networks. Jan 2016, 67. doi: 10.1146/annurev-psych-122414-033634.

Srivastava, N. – Hinton, G. E. – Krizhevsky, A. – Sutskever, I. – Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 2014, 15, 1, p. 1929–1958.

Stahlberg, F. – Byrne, B. On NMT Search Errors and Model Errors: Cat Got Your Tongue? 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. 3356–3362, Hong Kong, China, November 2019. Association for Computational Linguistics.

Straka, M. – Straková, J. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, p. 88–99, Vancouver, Canada, August 2017. Association for Computational Linguistics.

Straková, J. – Straka, M. – Hajič, J. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, p. 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

Sudarikov, R. – Mareček, D. – Kocmi, T. – Variš, D. – Bojar, O. CUNI submission in WMT17: Chimera goes neural. In Bojar, O. – Buck, C. – Chatterjee, R. – Federmann, C. – Graham, Y. – Haddow, B. – Huck, M. – Jimeno-Yepes, A. – Koehn, P. – Kreutzer, J. (Ed.) *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, p. 248–256. Association for Computational Linguistics, 2017.

Sun, F. – Ho, C. – Lee, H. LAMOL: LAnguage MOdeling for Lifelong Language Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Sundermeyer, M. – Schlüter, R. – Ney, H. LSTM neural networks for language modeling. In *Proc. Interspeech 2012*, p. 194–197, 2012.

Sussmann, H. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural Networks*. 1 1992, 5, 4, p. 589–593. ISSN 0893-6080.

Sutskever, I. – Vinyals, O. – Le, Q. V. Sequence to Sequence Learning with Neural Networks. In Ghahramani, Z. – Welling, M. – Cortes, C. – Lawrence, N. – Weinberger, K. Q. (Ed.) *Advances in Neural Information Processing Systems*, 27, p. 3104–3112. Curran Associates, Inc., 2014.

Szegedy, C. – Vanhoucke, V. – Ioffe, S. – Shlens, J. – Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 2818–2826, 2016.

Tan, X. – Ren, Y. – He, D. – Qin, T. – Zhao, Z. – Liu, T. Multilingual Neural Machine Translation with Knowledge Distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Tars, M. – Tättar, A. – Fisel, M. Extremely low-resource machine translation for closely related languages. In Dobnik, S. – Øvrelid, L. (Ed.) *Proceedings of the 23rd Nordic Conference on Computational Linguistics, NoDaLiDa 2021, Reykjavik, Iceland (Online), May 31 - June 2, 2021*, p. 41–52. Linköping University Electronic Press, Sweden, 2021a.

Tars, M. – Tättar, A. – Fišel, M. Extremely low-resource machine translation for closely related languages. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, p. 41–52, Reykjavik, Iceland (Online), May 31–2 June 2021b. Linköping University Electronic Press, Sweden.

Tarvainen, A. – Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In Guyon, I. – Luxburg, U. – Bengio, S. – Wallach, H. M. – Fergus, R. – Vishwanathan, S. V. N. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, p. 1195–1204, 2017.

Thillainathan, S. – Ranathunga, S. – Jayasena, S. Fine-Tuning Self-Supervised Multilingual Sequence-To-Sequence Models for Extremely Low-Resource NMT. In *2021 Moratuwa Engineering Research Conference (MERCon)*, p. 432–437, 2021.

Thompson, B. – Gwinnup, J. – Khayrallah, H. – Duh, K. – Koehn, P. Overcoming Catastrophic Forgetting During Domain Adaptation of Neural Machine Translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, p. 2062–2068, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Thrun, S. – Pratt, L. Y. (Ed.). *Learning to Learn*. Springer, 1998. ISBN 978-1-4613-7527-2.

Tiedemann, J. Parallel Data, Tools and Interfaces in OPUS. In Calzolari, N. – Choukri, K. – Declerck, T. – Dogan, M. U. – Maegaard, B. – Mariani, J. – Odijk, J. – Piperidis, S. (Ed.) *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, p. 2214–2218. European Language Resources Association (ELRA), 2012.

Tishby, N. – Zaslavsky, N. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop, ITW 2015, Jerusalem, Israel, April 26 - May 1, 2015*, p. 1–5. IEEE, 2015.

Tsai, Y.-H. H. – Bai, S. – Yamada, M. – Morency, L.-P. – Salakhutdinov, R. Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel. 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. 4344–4353, Hong Kong, China, November 2019. Association for Computational Linguistics.

Tsividis, P. – Pouncy, T. – Xu, J. L. – Tenenbaum, J. B. – Gershman, S. J. Human Learning in Atari. In *2017 AAAI Spring Symposia, Stanford University, Palo Alto, California, USA, March 27-29, 2017*. AAAI Press, 2017.

Bosch, A. Hidden Markov Models. In Sammut, C. – Webb, G. I. (Ed.) *Encyclopedia of Machine Learning and Data Mining*. New York, NY, USA: Springer, 2017. p. 609–611.

Variš, D. – Bojar, O. Unsupervised Pretraining for Neural Machine Translation Using Elastic Weight Consolidation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, p. 130–135, Florence, Italy, July 2019. Association for Computational Linguistics.

Variš, D. – Bojar, O. Sequence Length is a Domain: Length-based Overfitting in Transformer Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, p. 8246–8257, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

Vaswani, A. – Shazeer, N. – Parmar, N. – Uszkoreit, J. – Jones, L. – Gomez, A. N. – Kaiser, L. – Polosukhin, I. Attention is All you Need. In Guyon, I. – Luxburg, U. V. – Bengio, S. – Wallach, H. – Fergus, R. – Vishwanathan, S. – Garnett, R. (Ed.) *Advances in Neural Information Processing Systems 30*. San Francisco, CA, USA: Curran Associates, Inc., 2017. p. 6000–6010.

Vauquois, B. A survey of formal grammars and algorithms for recognition and transformation in mechanical translation. In Morrel, A. J. H. (Ed.) *Information Processing, Proceedings of IFIP Congress 1968, Edinburgh, UK, 5-10 August 1968, Volume 2 - Hardware, Applications*, p. 1114–1122, 1968.

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.

Wang, A. – Singh, A. – Michael, J. – Hill, F. – Levy, O. – Bowman, S. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, p. 353–355, Brussels, Belgium, November 2018a. Association for Computational Linguistics.

Wang, Y. – Zhang, J. – Zhai, F. – Xu, J. – Zong, C. Three Strategies to Improve One-to-Many Multilingual Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 2955–2960, Brussels, Belgium, October-November 2018b. Association for Computational Linguistics.

Ward, T. Structured Imagination: the Role of Category Structure in Exemplar Generation. *Cognitive Psychology*. 1994, 27, 1, p. 1–40. ISSN 0010-0285.

Williams, R. J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* 1992, 8, p. 229–256.

Williams, R. J. – Zipser, D. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*. 1989, 1, p. 270–280.

Wu, L. – Xia, Y. – Tian, F. – Zhao, L. – Qin, T. – Lai, J. – Liu, T. Adversarial Neural Machine Translation. In Zhu, J. – Takeuchi, I. (Ed.) *Proceedings of The 10th Asian Conference on Machine Learning, ACML 2018, Beijing, China, November 14-16, 2018*, 95 / *Proceedings of Machine Learning Research*, p. 534–549. PMLR, 2018.

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

Xie, S. – Girshick, R. B. – Dollár, P. – Tu, Z. – He, K. Aggregated Residual Transformations for Deep Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, p. 5987–5995. IEEE Computer Society, 2017.

Xiong, R. – Yang, Y. – He, D. – Zheng, K. – Zheng, S. – Xing, C. – Zhang, H. – Lan, Y. – Wang, L. – Liu, T. On Layer Normalization in the Transformer Architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, 119 / *Proceedings of Machine Learning Research*, p. 10524–10533. PMLR, 2020.

Xu, F. – Tenenbaum, J. B. Word Learning as Bayesian Inference. *Psychological Review*. 2007, p. 245–272.

Xu, H. – Durme, B. V. – Murray, K. W. BERT, mBERT, or BiBERT? A Study on Contextualized Embeddings for Neural Machine Translation. In Moens, M. – Huang, X. – Specia, L. – Yih, S. W. (Ed.) *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, p. 6663–6675. Association for Computational Linguistics, 2021.

Xu, K. – Ba, J. – Kiros, R. – Cho, K. – Courville, A. – Salakhudinov, R. – Zemel, R. – Bengio, Y. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In Bach, F. – Blei, D. (Ed.) *Proceedings of the 32nd International Conference on Machine Learning*, 37 / *Proceedings of Machine Learning Research*, p. 2048–2057, Lille, France, 07–09 Jul 2015. PMLR.

Yamagishi, H. – Kanouchi, S. – Sato, T. – Komachi, M. Controlling the Voice of a Sentence in Japanese-to-English Neural Machine Translation. In Nakazawa, T. – Mino, H. – Ding, C. – Goto, I. – Neubig, G. – Kurohashi, S. – Riza, I. H. – Bhattacharyya, P. (Ed.) *Proceedings of the 3rd Workshop on Asian Translation, WAT@COLING 2016, Osaka, Japan, December 2016*, p. 203–210. The COLING 2016 Organizing Committee, 2016.

Yu, X. – Zhang, H. – Song, Y. – Song, Y. – Zhang, C. What You See is What You Get: Visual Pronoun Coreference Resolution in Dialogues. 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. 5123–5132, Hong Kong, China, November 2019. Association for Computational Linguistics.

Zaheer, M. – Guruganesh, G. – Dubey, K. A. – Ainslie, J. – Alberti, C. – Ontanon, S. – Pham, P. – Ravula, A. – Wang, Q. – Yang, L. – Ahmed, A. Big Bird: Transformers for Longer Sequences. In Larochelle, H. – Ranzato, M. – Hadsell, R. – Balcan, M. F. – Lin, H. (Ed.) *Advances in Neural Information Processing Systems*, 33, p. 17283–17297. Curran Associates, Inc., 2020.

Zenke, F. – Poole, B. – Ganguli, S. Continual Learning Through Synaptic Intelligence. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, p. 3987–3995, 2017.

Zhang, B. – Williams, P. – Titov, I. – Sennrich, R. Improving Massively Multilingual Neural Machine Translation and Zero-Shot Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 1628–1639, Online, July 2020. Association for Computational Linguistics.

Zhang, B. – Bapna, A. – Sennrich, R. – Firat, O. Share or Not? Learning to Schedule Language-Specific Capacity for Multilingual Translation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a.

Zhang, C. – Bengio, S. – Hardt, M. – Recht, B. – Vinyals, O. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Zhang, C. – Bengio, S. – Hardt, M. – Recht, B. – Vinyals, O. Understanding Deep Learning (Still) Requires Rethinking Generalization. *Commun. ACM*. feb 2021b, 64, 3, p. 107–115. ISSN 0001-0782.

Zhang, Y. – Yang, Q. A Survey on Multi-Task Learning. *CoRR*. 2017, abs/1707.08114.

Zhu, Z. – Wu, J. – Yu, B. – Wu, L. – Ma, J. The Anisotropic Noise in Stochastic Gradient Descent: Its Behavior of Escaping from Sharp Minima and Regularization Effects. In Chaudhuri, K. – Salakhutdinov, R. (Ed.) *Proceedings of the 36th International Conference on Machine Learning*, 97 / *Proceedings of Machine Learning Research*, p. 7654–7663. PMLR, 09–15 Jun 2019.

Zhuang, F. – Qi, Z. – Duan, K. – Xi, D. – Zhu, Y. – Zhu, H. – Xiong, H. – He, Q. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*. 2021, 109, 1, p. 43–76.

List of Abbreviations

AGI artificial general intelligence. 2, 35, 65
AR autoregressive. 21, 22, 24

BLEU bilingual evaluation understudy. 32, 45–48, 55–57, 61, 62, 64, 85, 103, 105, 118, 119, 121, 122
BoW bag of words. 93
BPE byte-pair encoding. 3, 40, 45, 54, 59, 61, 84
BPTT backpropagation through time. 23

CE cross-entropy. 31
CF catastrophic forgetting. 2, 3, 5–7, 9, 11, 18–20, 65–67, 69, 76, 77, 83, 86, 114, 115
CI catastrophic interference. 18, 65
CIFAR-10 Canadian Institute For Advanced Research. 2
CNN convolutional neural network. 26, 27, 31
COMET Crosslingual Optimized Metric for Evaluation of Translation. 85, 103

DAN deep averaging network. 92
DL deep learning. 1, 2, 18, 35, 36, 54

ELBO evidence lower bound. 92
EM expectation-maximization. 92
EOS end-of-sequence. 24, 33, 50, 52, 63
EWC elastic weight consolidation. 6, 9–12, 67, 69, 71–88, 114, 115, 117, 119, 120

FFN feed-forward network. 10, 26, 30, 95, 96, 102, 108, 111
FI Fisher information. 9, 79–81, 83
FIM Fisher information matrix. 68, 73, 78–82, 84–87

GLUE General Language Understanding Evaluation. 39
GRU gated recurrent unit. 23, 42

HMM hidden Markov model. 21

IDF inverse document frequency. 55
IID independent and identically distributed. 38, 39
IL incremental learning. 6, 7, 11, 13, 14, 18–20, 67, 77, 78, 87, 115

IWSLT International Conference on Spoken Language Translation. 72, 73

KL Kullback–Leibler. 57, 58

LDD long-distance dependencies. 4, 41
LM language model. 6, 12, 16, 19, 37, 39, 55–57, 70–76, 87, 90
LSTM long short-term Memory. 23, 26, 42

MAP maximum a posteriori. 76
MEE maximum entropy estimation. 70
MFFN masked feed-forward network. 96, 97
MHA multi-head attention. 3, 10, 29, 30, 89, 95, 111
ML machine learning. 35
MLM masked language model. 16, 74
MLP multi-layered perceptron. 1, 76
MMHA masked multi-head attention. 95–97
MNIST Modified National Institute of Standards and Technology. 2, 76
MoE mixture-of-experts. 90
MRT minimum risk training. 32
MSCOCO Microsoft Common Objects in Context. 40
MT machine translation. 2, 16, 41, 42, 44, 47, 50, 52, 54, 55, 70, 71, 74
MTL multi-task learning. 3, 6, 11, 13–16, 18–20, 66, 83, 90

NAR non-autoregressive. 22
NLI natural language inference. 16
NLL negative log-likelihood. 31, 32, 50
NLP natural language processing. 2–4, 8, 11, 14–16, 19, 22, 31, 36, 38–40, 42, 44, 63, 96
NMT neural machine translation. 3, 5–20, 31–33, 39, 40, 59, 63, 69–76, 78, 83, 85–88, 90, 95, 96, 102, 103, 106, 114, 115
NN neural network. 1–3, 22, 23, 27, 33, 38, 67

OOV out-of-vocabulary. 5, 20, 60

PBMT phrase-based machine translation. 70
PI path integral. 69

QA question answering. 16

RBF radial basis function. 42
ReLU rectified linear unit. 30
RL reinforcement learning. 32, 76, 92

RNN recurrent neural network. 23, 24, 26–28, 31–33, 41, 42
ROUGE Recall-Oriented Understudy for Gisting Evaluation. 32

SGD stochastic gradient descent. 38, 120
SMT statistical machine translation. 70
SoTA state-of-the-art. 1–3, 11, 15, 35, 36, 39, 65, 76
SQuAD Stanford Question Answering Dataset. 39
STE straight-through estimator. 94, 97, 110

TED Technology, Entertainment and Design. 72
TER Translation Error Rate. 120
TF-IDF term frequency-inverse document frequency. 54–57, 63

WMT Conference on Machine Translation. 39

List of Tables

List of Figures

List of Publications

Variš, D. – Bojar, O. Sequence Length is a Domain: Length-based Overfitting in Transformer Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, p. 8246–8257, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics

- This paper presents the experiments probing the length-overfitting in Transformer models. This paper is the basis of the length-overfitting experiments in Chapter 3
- Citations (without self-citations): 9

Variš, D. – Bojar, O. Unsupervised Pretraining for Neural Machine Translation Using Elastic Weight Consolidation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, p. 130–135, Florence, Italy, July 2019. Association for Computational Linguistics

- This paper introduces the work on unsupervised pretraining of low-resource NMT models using monolingual corpora and fine-tuning using EWC regularization. This paper is, in part, presented in Chapter 5.
- Citations (without self-citations): 12

Kocmi, T. – Variš, D. – Bojar, O. CUNI Basque-to-English Submission in IWSLT18. In Turchi, M. – Niehues, J. – Federico, M. (Ed.) *Proceedings of the International Workshop on Spoken Language Translation*, p. 142–146, Karlsruhe, Germany, 2018. University of Brugge, Karlsruhe Institute of Technology

- This paper describes the submission to the IWSLT18 Basque-to-English low-resource translation task.
- Citations (without self-citations): 1

Helcl, J. – Libovický, J. – Variš, D. CUNI System for the WMT18 Multimodal Translation Task. In Bojar, O. (Ed.) *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Tasks*, 2, p. 622–629, Stroudsburg, PA, USA, 2018b. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-948087-81-0

- This paper describes the submission of the multi-modal Transofmer model to WMT18 Multi-modal Translation Task
- Citations (without self-citations): 73

Helcl, J. – Libovický, J. – Kocmi, T. – Musil, T. – Cífka, O. – Variš, D. – Bojar, O. Neural Monkey: The Current State and Beyond. In Neubig, G. – Cherry, C. (Ed.) *The 13th Conference of The Association for Machine Translation in the Americas, Vol. 1: MT Researchers' Track*, p. 168–176, Stroudsburg, PA, USA, 2018a. The Association for Machine Translation in the Americas, The Association for Machine Translation in the Americas

- This paper describes the updated version of the Neural Monkey neural sequence learning framework.
- Citations (without self-citations): 10

Bojar, O. – Kocmi, T. – Mareček, D. – Sudarikov, R. – Variš, D. CUNI Submission in WMT17: Chimera Goes Neural. In Bojar, O. (Ed.) *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, 2, p. 248–256, Stroudsburg, PA, USA, 2017. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-945626-96-8

- This paper describes the submission of the hybrid NMT system Chimera to WMT17 News Translation task. Neural models were implemented in the Neural Monkey.
- Citations (without self-citations): 3

Bojar, O. – Dušek, O. – Kocmi, T. – Libovický, J. – Novák, M. – Popel, M. – Sudarikov, R. – Variš, D. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In Sojka, P. – Horák, A. – Kopeček, I. – Pala, K. (Ed.) *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*, no. 9924 in Lecture Notes in Computer Science, p. 231–238, Cham / Heidelberg / New York / Dordrecht / London, 2016. Masaryk University, Springer International Publishing. ISBN 978-3-319-45509-9

- This paper describes a release of English-Czech parallel corpus CzEng 1.6, the previous version of CzEng 2.0 used in the thesis.
- Citations (without self-citations): 89

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): 197