Multi-domain Neural Machine Translation

Minh-Quang Pham

université PARIS-SACLAY

Multi-domain Neural Machine Translation

Traduction Automatique Neuronale Multidomaine

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)

Spécialité de doctorat: Informatique

Unité de recherche: Université Paris-Saclay, CNRS, Laboratoire interdisciplinaire des sciences du numérique, 91405, Orsay, France

Graduate School : Informatique et sciences du numérique

Référent : Faculté des sciences d'Orsay

Thèse soutenue à Paris-Saclay, le 10 décembre 2021, par

Minh-Quang PHAM

Composition du jury

Pierre ZWEIGENBAUM	Président
Directeur de recherche, LISN/CNRS, Université Paris-Saclay	
Rico SENNRICH	Rapporteur & Examinateur
Professeur, University of Zurich	
Alexander M. FRASER	Rapporteur & Examinateur
Professeur, Ludwig Maximilian University of Munich	
Marine CARPUAT	Examinatrice
Professeur adjoint, University of Maryland	

Direction de la thèse

François YVON	Directeur de thèse
Directeur de recherche, LISN/CNRS, Université Paris-Saclay	
Josep CREGO	Co-encadrant
Chercheur, SYSTRAN	

THESE DE DOCTORAT

NNT : 2021UPASG109

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Traduction Automatique Neuronale Multidomaine

Mots clés: Traduction neuronale, Adaptation au domaine, Apprentissage multi-tâche

Résumé: Aujourd'hui, les systèmes de traduction automatique neuronale (TAN) constituent des systèmes de pointe en traduction automatique (TA). Cependant, ces modèles de traduction nécessitent des données d'entraînement relativement volumineuses et ont de la difficulté à traduire des textes dans les domaines spécifiques. Un domaine peut être constitué de textes d'un sujet particulier ou de textes écrits dans un but particulier. Bien que les systèmes TAN puissent être adaptés pour une meilleure qualité de traduction dans un domaine cible étant donné un corpus d'apprentissage représentatif, cette technique a des effets secondaires négatifs, notamment une fragilité contre des exemples hors domaine et un "oubli catastrophique" des domaines précédents représentés dans les données d'entraînement. De plus, un système de traduction doit faire face à de nombreux domaines dans des applications réelles, ce qui rend impossible d'apprendre un modèle par un domaine. Une solution prometteuse consiste à construire des systèmes TAN multidomaines formés à partir des données de nombreux domaines et adaptés à plusieurs domaines cibles. Il y a deux motivations. Premièrement, les grands corpus d'apprentissage améliorent la généralisation du système TAN. Deuxièmement, les textes d'un domaine peuvent être utiles pour adapter un modèle TAN à un domaine similaire. La pénurie des données et l'effet de transfert positif hypothétique sont également deux raisons principales pour le développement des systèmes TAN multilingues. Maintenir plusieurs systèmes de traduction automatique bilingues nécessite de nombreuses ressources matérielles, car le nombre de paires de langues augmente de façon quadratique avec l'augmentation du nombre de langues. Les systèmes TAN multidomaines et multilingues sont essentiels pour économiser des ressources pour l'industrie TA et améliorer la qualité du service TA. Cette thèse unifie d'abord l'adaptation au domaine et l'adaptation multidomaines dans un cadre mathématique. De plus, nous passons en revue la littérature sur l'adaptation multidomaines à travers une approche structurelle en distinguant quatre situation d'adaptation et en associant les méthodes proposées à chaque cas d'application. Deuxièmement, nous proposons une nouvelle évaluation multicritères des approches multidomaines. Nous soulignons les exigences d'un système multidomaines et réalisions une comparaison exhaustive d'un large ensemble de méthodes. Nous proposons également de nouvelles méthodes pour l'adaptation multidomaines, y compris les plongements de mot parcimonieux, les couches parcimonieuse et les adaptateurs résiduels, qui sont relativement légers et capables d'adapter un TAN modèle à de nombreux domaines. Pour équilibrer l'hétérogénéité des données d'entraînement, nous explorons et étudions les techniques à l'échantillonnage dynamique des données, qui adaptent de manière itérative la distribution de l'entraînement à une distribution de test prédéterminée. Enfin, nous nous intéressons aux approches de traduction avec des contextes augmentés, qui réutilisent des mémoires de traduction similaires pour améliorer la prédiction d'une phrase. Nous analysons et comparons plusieurs méthodes de cette famille et démontrons qu'elles conviennent pour adapter notre système TAN à un domaine inconnu au détriment de coûts de calcul supplémentaires.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Multi-domain Neural Machine Translation

Keywords: Neural machine translation, Domain adaptation, multi-task learning

Abstract: Today, neural machine translation (NMT) systems constitute state-of-the-art systems in machine translation. However, such translation models require relatively large train data and struggle to handle a specific domain text. A domain may consist of texts from a particular topic or texts written for a particular purpose. While NMT systems can be adapted for better translation quality in a target domain given a representative train corpus, this technique has adverse side-effects, including brittleness against out-of-domain examples and "catastrophic forgetting" of previous domains represented in the train data. Moreover, one translation system must cope with many possible domains in real applications, making the "one domain one model" impractical. A promising solution is to build multi-domain NMT systems trained from many domains and adapted to multiple target domains. The rationale behind this is twofold. First, large train corpora improve the generalization of the NMT system. Secondly, texts from one domain can be valuable for adapting an NMT model to a similar domain. The scarcity of data and the hypothetical positive transfer effect are also two main reasons for building multilingual NMT systems. Maintaining multiple bilingual MT systems requires lots of hardware resources as the number of language pairs grows quadratically with the increasing number of languages. Both multi-domain and multilingual NMT systems are essential for saving resources for the MT industry and improving the quality of the MT service.

This thesis first unifies domain adaptation and multi-domain adaptation in one mathematical framework. In addition, we review the literature of (multi-)domain adaptation through a structural approach by pointing out four principal cases and matching previous methods to each application case. Secondly, we propose a novel multi-criteria evaluation of multi-domain approaches. We point out the requirements for a multi-domain system and perform an exhaustive comparison of a large set of methods. We also propose new methods for multi-domain adaptation, including sparse word embeddings, sparse layers, and gated residual adapters, which are cheap and able to handle many domains. To balance the heterogeneity in the train data, we explore and study techniques relating to dynamic data sampling, which iteratively adapt the train distribution to a predetermined testing distribution. Finally, we are interested in context augmented translation approaches, which reuse similar translation memories to improve the prediction of a sentence. We carefully analyze and compare several methods in this line and demonstrate that they are suitable for adapting our NMT system to an unknown domain at the expense of additional computational costs.

Acknowledgments

This thesis would not have been made possible without the continuous support and help from my Ph.D. supervisor François Yvon and my Ph.D. co-supervisor Josep Maria Crego. They spent countless hours discussing research with me, helping me with paper writing, and were very supportive throughout my Ph.D. The outstanding critical thinking of François will influence me for the rest of my research career. I have also learned a lot from Josep's problem-solving skills. I consider myself incredibly fortunate to have been their student. My second thank goes to all the members of the jury committee, Alexander Fraser, Marine Carpuat, Sennrich Rico, who gave me many insights during the defense and also helped me correct many errors in my thesis.

Then, I would like to thank my parents and my little brother. Without their emotional support, I could not have gone this far in my research career. I want to thank also my long-time girlfriend, who also is doing a Ph.D. thesis, for having been very supportive during my Ph.D. and for her helpful outside perspectives.

Next, I would like to thank my colleagues in Systran: Guillaume Klein, Natalia Segal, Mickael Mani, and Elise Michon for many running sessions and Dakun Zhang for many interesting research papers. I will always cherish the race Paris-Versailles 2018. My thank also goes to Jean Senellart, CEO of Systran, for helping to create this collaborative project between Systran and LISN.

Besides, I would like to thank many friends at LISN, Aina Gari-Soler, François Buet, Alban Petit, Aman Berhe, Jitao Xu, Paul Lerner, Marc Benzahra, Shu Okabe and Anh Khoa Ngo Ho. I always cherish our coffee breaks.

This work was generously granted access to the HPC resources of [TGCC/CINES/IDRIS] under the allocation 2020- [AD011011270] made by GENCI (Grand Equipement National de Calcul Intensif).

Abstract

Today, neural machine translation (NMT) systems constitute state-of-the-art systems in machine translation. However, such translation models require relatively large train data and struggle to handle a specific domain text. A domain may consist of texts from a particular topic or texts written for a particular purpose. While NMT systems can be adapted for better translation quality in a target domain given a representative train corpus, this technique has adverse side-effects, including brittleness against out-of-domain examples and "catastrophic forgetting" of previous domains represented in the train data. Moreover, one translation system must cope with many possible domains in real applications, making the "one domain one model" impractical. A promising solution is to build multi-domain NMT systems trained from many domains and adapted to multiple target domains. The rationale behind this is twofold. First, large train corpora improve the generalization of the NMT system. Secondly, texts from one domain can be valuable for adapting an NMT model to a similar domain. The scarcity of data and the hypothetical positive transfer effect are also two main reasons for building multilingual NMT systems. Maintaining multiple bilingual MT systems requires lots of hardware resources as the number of language pairs grows quadratically with the increasing number of languages. Both multi-domain and multilingual NMT systems are essential for saving resources for the MT industry and improving the quality of the MT service.

This thesis first unifies domain adaptation and multi-domain adaptation in one mathematical framework. In addition, we review the literature of (multi-)domain adaptation through a structural approach by pointing out four principal cases and matching previous methods to each application case. Secondly, we propose a novel multi-criteria evaluation of multi-domain approaches. We point out the requirements for a multi-domain system and perform an exhaustive comparison of a large set of methods. We also propose

new methods for multi-domain adaptation, including sparse word embeddings, sparse layers, and gated residual adapters, which are cheap and able to handle many domains. To balance the heterogeneity in the train data, we explore and study techniques relating to dynamic data sampling, which iteratively adapt the train distribution to a predetermined testing distribution. Finally, we are interested in context augmented translation approaches, which reuse similar translation memories to improve the prediction of a sentence. We carefully analyze and compare several methods in this line and demonstrate that they are suitable for adapting our NMT system to an unknown domain at the expense of additional computational costs.

Contents

Acknowledgment **4**

Abstract **5**

List of Figures **13**

List of Tables **15**

List of Abbreviations **20**

1 Introduction **22**

1.1 Motivation 22

1.2 Contributions 23

1.3 Outline 24

2 Neural Machine Translation: a review **25**

2.1 Text preprocessing for NMT 25

2.1.1 Word tokenization 26

2.1.2 Subword tokenization 26

2.1.3 Character tokenization 27

2.1.4 Byte-level tokenization 28

2.2 NMT's main components 28

2.3 Recurrent neural machine translation 32

2.3.1 GRU, LSTM layers 32

2.3.2 RNN encoders 33

2.3.3 RNN decoders 34

2.4 Convolutional neural machine translation 35

2.4.1 Convolutional encoders 36

2.4.2 Convolutional decoders 36
2.4.3 Positional embeddings 37
2.5 Attention-based neural machine translation 37
2.5.1 Transformer encoders 37
2.5.2 Transformer decoders 38
2.5.3 Positional embeddings 39
2.6 Training NMT models 39
2.6.1 Tips and tricks in training an NMT model 40
2.7 Inference with an NMT model 41
2.8 MT evaluation 42

3 (Multi-)domain adaptation in neural machine translation 43
3.1 What is a domain? 44
3.2 Machine translation (multi-)domain adaptation - a multi-faceted problem . 46
3.2.1 From domain adaptation MT to multi-domain adaptation MT . . . 46
3.2.2 Four main sub-problems 47
3.3 Supervised (multi-)domain adaptation 49
3.3.1 Model-centric approaches 49
3.3.2 Data-centric approaches 52
3.4 Undetermined train domain, determined test domain 53
3.4.1 Model-centric approaches 53
3.4.2 Data-centric approaches 56
3.5 Determined train domain, undetermined test domain 59
3.5.1 Model-centric approaches 60
3.5.2 Data-centric approaches 61
3.6 Undetermined train domain, undetermined test domain 61
3.6.1 Model-centric approaches 61
3.6.2 Data-centric approaches 61
3.7 Multi-domain and multi-lingual machine translation 61
3.8 Conclusions 62

4 Revisiting supervised multi-domain machine translation 64

4.1 Introduction . . . 64

4.2 Requirements of multi-domain MT . . . 65

4.2.1 Formalizing multi-domain translation . . . 65

4.2.2 Reasons for building MDMT systems . . . 66

4.3 Challenging multi-domain systems . . . 68

4.3.1 Multi-domain systems should be effective . . . 68

4.3.2 Robustness to fuzzy domain separations . . . 68

4.3.3 Scaling to a large number of domains . . . 69

4.4 Experimental settings . . . 70

4.4.1 Data and metrics . . . 70

4.4.2 Baselines . . . 71

4.4.3 Multi-domain systems . . . 72

4.5 Results and discussion . . . 73

4.5.1 Computational costs . . . 73

4.5.2 Performance of MDMT systems . . . 73

4.5.3 Redefining domains . . . 75

4.5.4 Automatic domains . . . 78

4.6 Conclusions and outlook . . . 78

5 Lexicalized domain representations and Contextualized domain representation 80

5.1 Introduction . . . 80

5.2 Lexicalized domain representations . . . 81

5.2.1 Multi-domain machine translation . . . 81

5.2.2 Lexicalized domain embeddings . . . 82

5.3 Contextualized domain representations . . . 83

5.4 Experiments . . . 83

5.4.1 Domains, data and metrics . . . 83

5.4.2 Baselines . . . 85

5.4.3 Implementing Lexicalized Domain Representations . . . 86

5.4.4 Implementing Contextualized domain representations 87
5.4.5 Results 87
5.5 Complementary experiments 91
5.5.1 Balancing domain-agnostic and domain-specific representations in LDR 91
5.5.2 Additional domain setting with LDR 92
5.5.3 Analysis of Word Embeddings of LDR 93
5.6 Conclusions and outlook 95

6 Multi-domain residual adapters 96
6.1 Introduction 96
6.2 Residual adapters 97
6.2.1 Basic architecture 97
6.2.2 Highway Residual Adapters 99
6.2.3 Gated Residual Adapters 99
6.3 Experimental settings 101
6.3.1 Data and metrics 101
6.3.2 Baseline architectures 101
6.3.3 Multi-domain systems 102
6.3.4 Varying the positions and number of residual adapters 104
6.3.5 Regularizing adapter tuning 104
6.3.6 Highway and Gated Residual Adapters 105
6.4 Conclusions and outlook 106

7 Dynamic sampling strategies for multi-domain adaptation 108
7.1 Introduction 108
7.2 Learning with multiple data sources 109
7.3 Multi-Domain Automated Curriculum 110
7.3.1 Basic principles 110
7.3.2 MDAC for (multi) domain adaptation 112
7.4 Experimental settings 112

7.4.1 Data and metrics . . . 112
7.4.2 Baseline systems . . . 112
7.4.3 CL and DDS's re-implementation . . . 113
7.4.4 MDAC systems . . . 113
7.4.5 Experimental tasks . . . 114
7.5 Results and discussion . . . 115
7.5.1 Domain Adaptation . . . 115
7.5.2 Bi-domain adaptation . . . 115
7.5.3 Multi-domain adaptation . . . 116
7.5.4 Unseen domain . . . 117
7.5.5 Automatic clustering . . . 118
7.5.6 Reward analysis . . . 118
7.6 Related Work . . . 119
7.7 Conclusions and outlook . . . 119

8 Attempts at unsupervised multi-domain adaptation 121
8.1 Introduction . . . 121
8.2 Context-augmented NMT . . . 122
8.2.1 Similarity Computation . . . 122
8.2.2 How to present similar translations to an NMT model . . . 123
8.3 Experimental Framework . . . 125
8.3.1 Corpora . . . 125
8.3.2 System Configurations . . . 126
8.4 Results . . . 127
8.5 Discussion . . . 131
8.6 Conclusions and outlook . . . 132

9 Conclusions 133

Bibliography 136

A A - Description of multi-domain systems (Chapter 4) 155

B B - Experiments with continual learning (Chapter 4) 157

C C - Experiments with automatic domains (Chapter 4) 158

D D - Generalized Multi-Domain Dynamic Adaptation Curriculum Algorithm (Chapter 7) 160

E E - Experiments with automatic domains (Chapter 7) 161

F F - Résumé en Français 164

List of Figures

2.1 Illustration of context range at each token in different encoding mechanism 30

2.2 Illustration of 3 most popular auto-regressive decoding paradigms 31

2.3 Illustration of 3 most popular multi-layer auto-regressive decoding paradigms 32

3.1 Training and testing with distribution mismatch 46

3.2 Model-centric's brief overview 53

3.3 A complete overview of multi-domain adaptation with the four primary settings and the two groups of approaches. 62

4.1 Ability to handle a new domain 77

5.1 Lexicalized domain embeddings. When processing a sample from domain 2, we only activate the corresponding parameter region (θ_2) in the input embeddings; the remaining domain-specific parts are zeroed out and do not receive any update. The domain-agnostic part is always active and is updated irrespective of the input domain. 82

5.2 Contextualized domain representations. When processing a sample from domain 1, only the signal of the domain-agnostic nodes and the 1^{th} domain's nodes are passed to higher layers, other nodes are dropped out. . . 84

6.1 Highway residual adapter network 99

7.1 Training and testing with distribution mismatch. We consider just three domains, and represent vectors of mixture weights $\boldsymbol{\lambda}^s$ and $\boldsymbol{\lambda}^t$ in the 3-dimensional simplex. Training with weights in (a) and testing with weights in (c) is supervised multi-source domain adaptation to domain 2 (d_2), while (b)-(c) is the unsupervised version, with no training data from d_2; training with weights in (a) and testing with weights in (d) is multi-domain learning, also illustrated with configurations (a)-(e) (training domain d_1 is not seen in test), and (b)-(d) (test domain d_2 is unseen in training). 110
7.2 Evolution of the sampling distribution during training. 117
7.3 Evolution of the rewards during training. 118

List of Tables

4.1 Corpora statistics: number of parallel lines ($\times 10^3$) and proportion in the basic domain mixture (which does not include the NEWS domain), number of tokens in English and French ($\times 10^6$), number of types in English and French ($\times 10^3$), number of types that only appear in a given domain ($\times 10^3$). MED is the largest domain, containing almost 70% of the sentences, while REL is the smallest, with only 3% of the data. 71

4.2 The $\mathscr{H}$-divergence between domains 71

4.3 Translation performance of MDMT systems based on the same Transformer (top) or RNN (bottom) architecture. The former contains 65m parameters, the latter has 51m. For each system, we report the number of additional domain specific parameters, BLEU scores for each domain, domain-weighted (WAVG) and unweighted (AVG) averages. For weighted-averages, we take the domain proportions from Table 4.1. Boldface denotes significant gains with respect to Mix-Nat (or Mix-Nat-RNN, for WDCMT), underline denotes significant losses. 74

4.4 Translation performance with variable domain definitions. In the *Split*/*Merge* experiments, we report BLEU differences for the related test set(s). In the test NEW, we report BLEU differences between each MDMT system and the generic system Mixed-Nat. In the test RND, we report BLEU differences between using random domain tag and using true domain tag. Underline denotes significant loss when domains are changed wrt. the baseline situation; bold for a significant improvement over FT-Full; (*) MDMT systems ignoring test domains. 75

4.5 BLEU scores computed by merging the 10 smaller, medium, and larger cluster test sets. Best score for each group is in boldface. For the small clusters, full-fine tuning is outperformed by several MDMT systems - see details in Appendix C. 78

4.6 Translation performance with automatic domains, computed with the original test sets. Significance tests are for comparisons with the 6-domain scenario (Table 4.3). 79

5.1 Corpora statistics. 84

5.2 BLEU scores for Transformer systems . Boldface denotes significant gains with respect to Mixed − Nat. 89

5.3 BLEU scores for RNN systems 90

5.4 BLEU scores for RNN systems. Comparison between WDCMT and LDR_{pred} built using conditional GRUs. 90

5.5 Translation performance of MDMT systems based on the same Transformer architecture. The former contains 65m parameters. For each system, we report the number of additional domain specific parameters, BLEU scores for each domain, domain-weighted (WAVG) and unweighted (AVG) averages. For weighted-averages, we take the domain proportions from Table 4.1. Boldface denotes significant gains with respect to Mix-Nat, underline denotes significant losses. 91

5.6 BLEU scores for the Transformer architecture for varying domain-specific embedding sizes 92
5.7 BLEU scores for the Transformer architecture when including IT as additional domain 92
5.8 Analyzing the variation of embeddings across domains. For each word or subword we also report the size of the intersection (between 0 and 10). . . 94

6.1 Corpora statistics for En→De: number of parallel lines ($\times 10^3$) and proportion in the basic domain mixture. WEB is the largest domain, containing about 54% of the sentences, while BANK and TOUR are very small. 101
6.2 Translation performance of various multi-domain MT systems (En→Fr) compared to variants of the residual adapter models. The performance of MDMT contrasts is borrowed from Table 4.3. 103
6.3 Translation performance of various fine-tuned systems (En→Fr). We report BLEU scores for each domain, as well as averages across domains. Column PARAMS reports the number of domain-agnostic/domain-specific parameters. 105
6.4 Translation performance of various fine-tuned systems (En→De). We report BLEU scores for each domain, as well as averages across domains. Column PARAMS reports the number of domain-agnostic/domain-specific parameters. 105
6.5 Translation performance of highway and gated variants for En→Fr. NEWS is excluded from the training data and considered as an out-of-domain test. 106

7.1 Domain adaptation experiments. We report BLEU scores of each method for 6 target domains and their average: each column corresponds to a distinct system. (*) MDAC is significantly better than CL, fine-tuning and DDS with $p < 0.05$. (**) MDAC is significantly better than CL and DDS with $p < 0.05$. (***) MDAC is significantly better than CL, fine-tuning with $p < 0.05$. 115

7.2 Adapting to two domains. For a given line, non empty columns correspond to the pair of target domains. (*) MDAC is significantly better than DDS with $p < 0.05$. 116

7.3 Multi domain adaptation. For a given line, all the columns *correspond to the same multi-domain system*. (*) MDAC is significantly better than `Mixed-`α with $p < 0.05$. (**) MDAC is significantly better than DDS with $p < 0.05$. 117

7.4 Unseen domain adaptation (left) and unsupervised adaptation (right). For a given line, each column corresponds to one distinct system. (*) MDAC is significantly better than DDS. 117

8.1 Corpora statistics. Note that K stands for thousands and L_{mean} is the average length in words. 125

8.2 Efficiency (tokens/second) of each step for different inference configurations. All steps run on CPU (16 cores). K stands for thousands. 128

8.3 `BLEU` scores for various model configurations and 8 test domains. Smaller numbers correspond to the number of input sentences in each domain for which at least one similar sentence is found. 128

8.4 Translation performance for the `NEWS` and `WIKI` domain test sets using similar sentences retrieved from parallel data (par) and from both parallel and monolingual (par+mon) data. The first two rows correspond to experiments already shown in Table 8.3. 130

8.5 Number of unrelated words appearing in test sets according to different augmentation schemes. The last row indicates the total number of unrelated words included in 1-best `FM` similar sentences. 131

8.6 Results for a reduced test set (279 sentences) using `CBON` when integrating human and synthetic (back-translated) translations. Here, $par = parallel$, $mon = monolingual$ 132

B.1 Ability to handle a new domain. We report BLEU scores for a complete training session with 7 domains, as well as differences with (left) training with 6 domains (from Table 4.3); (right) continuous training mode. Averages only take into account six domains (NEWS excluded). Underline denotes a significant loss, bold a significant gain. 157

C.1 Complete results for the experiments with automatic domains. For each cluster, we report: the majority domain when one domain accounts for more than 75% of the class; training and test sizes; and BLEU scores obtained with the various systems used in this study. Most test sets are too small to report significance tests. 159

E.1 Automatic clustering experiments. We report the size of each cluster. In the 6 left columns, each line gives the proportions of the domains in each cluster. In the 6 right columns, each column corresponds to a MDAC experiment; each line gives the cumulated proportion of the corresponding cluster in the training data. For instance, when targeting the domain ECB, cluster 4 (mostly LAW) is sampled with a probability of 0.07, and cluster 16 (mostly ECB) is sampled with probability 0.08. For each system, we underline the most often sampled clusters. 163

List of Abbreviations

Σ_x	source vocabulary
Σ_y	tgt vocabulary
ANMT	Attention-based neural machine translation
BERT	Bidirectional Encoding Representation Transformer
BOS	begin-of-sentence
BPE	Byte Pair Encoding
BPTT	Back-propagation through time
CDR	Contextualized domain representation
CNMT	Convolutional neural machine translation
CNN	Convolutional neural network
EOS	end-of-sentence
EWC	Elastic weight consolidation
FDA	Feature Decay Algorithm
GRU	Gated recurrent unit
LDR	Lexicalized domain representation
LHUC	Learning hidden unit contribution
LM	Language modeling or Language model
LSTM	Long-short term memory
MDMT	Multi-domain machine translation
MLE	Maximum likelihood estimation
NLM	Neural Language Model
NMT	Neural machine translation
OOV	Out of vocabulary
RBMT	Rule-based Machine Translation
RNMT	Recurrent neural machine translation
RNN	Recurrent neural network
SGD	Stochastic gradient descent

SMT	Statistical machine translation
TF-IDF	Term Frequency-Inverse Document Frequency
TM	Translation memory

Chapter 1

Introduction

1.1 Motivation

A neural machine translation (NMT) model usually has trouble translating sentences that differ in genre, register, or theme from the sentences used for training the model. This is a common limitation of data-driven machine learning methods, whose performance is guaranteed by assuming that the training and testing distributions are identical. Therefore, to achieve high performance in a given domain, we must carefully tailor the NMT model to that domain. The problem of tailoring an NMT model to a target domain is referred to as the **domain adaptation problem**. Two factors make this problem complex, including the scarcity of training data from the target domain and the catastrophic forgetting problem of the deep models. The lack of training data drives us to leverage parallel data from other domains to train our NMT models. The neural network-based models need many data to optimize their parameters. Therefore, we usually have to adapt our NMT model to the target domain using lots of out-of-domain data and a small amount of data from the target domain. Second, several approaches to adapting an NMT model by finetuning it with the in-domain data only make its performance very brittle to the out-of-domain test. This problem is referred to as **catastrophic forgetting** in the neural network literature. The neural models tend to perform dramatically worse in previous tasks after being trained to perform their current tasks. In real applications, we usually aim to significantly improve the performance in the target domain and the robustness of NMT models with respect to previous training domains.

The domain problem is addressed to some extent by domain adaptation approaches as long as the testing domain is known before training. However, in many applications, such as online translation on the web, the text to translate can be from any domain. We consider this situation ”non domain-deterministic testing” in our review 3. In this situation, we could build domain-expert models for the source domains and combine their predictions during the inference (Saunders et al., 2019) to get a domain-adapted translation on the fly. Besides the mixture of model paradigm, several methods use context to improve the translation of similar sentences. However, these methods do not guarantee domain

robustness against the out-of-domain text.

Moreover, an MT engine has to translate text from many domains whose genre and topic are highly variable in real applications. The strategy ”one domain / one model” will cost us largely when the number of domains explodes. Therefore, developing a multi-domain machine translation (MDMT) system is essential for the MT business.

This thesis aims to provide a complete overview of the multi-domain adaptation problem in machine translation and study the approaches to adapting an NMT system to many domains with a small computation and storage cost.

1.2 Contributions

In this thesis, our contributions are as follows.

First, we provide a generalized framework of the machine translation (multi-)domains adaptation problem. We point out four main situations in the domain mismatch problem. We provide a complete match between each case and its feasible methods.

Second, we provide a new multi-criteria evaluation for MT (multi-)domain methods. We reevaluate a large set of methods with our proposed experimental settings corresponding to our proposed criteria.

Thirdly, we propose, evaluate and analyze a cheap MT multi-domain method, which uses sparse word embedding with domain-specified units. The method is much cheaper than residual adapters. Besides an improvement in some mild multi-domain settings, the method can handle a growing number of domains. We extend the idea of sparse representation to higher layers of an NMT model. We demonstrate an equivalent performance of the method to several strong MDMT methods. We propose a novel analysis method for word embeddings, which identifies domain-agnostic and domain-specific tokens by observing the variation of K nearest neighbors of one token while changing its domain.

Our fourth contribution is a thorough study on the use of the residual adapters in (multi-)domain adaptation. We demonstrate its practicality and strong performance in a multi-domain MT setting consisting of a large set of domains with unbalanced sizes. We propose different regularization methods to avoid overfitting on low-resourced domains. Finally, we propose two more robust variants that are robust with respect to the domain label errors and slightly reduce the computation cost.

Next, we study dynamical sampling strategies for multi-domain machine translation. We show that those methods improve the data sampling from the mix of in-domain corpora with respect to the heuristic fixed sampling strategy. Furthermore, we demonstrate their effectiveness in several particular settings such as uni-domain adaptation, bi-domain adaptation, and unseen domain adaptation.

Finally, we study two popular paradigms for unknown test domains which rely on text retrieval. Those techniques search for the most similar translations and incorporate this additional information into the prediction of an NMT model. We demonstrate their efficacy and their weakness as well. Besides, we propose a simple variant that slightly improves the performance of previous techniques and is able to leverage synthesis translations.

1.3 Outline

The structure of the thesis is as follows.

Chapter 2 is a review of neural machine translation. The chapter provides basic knowledge in the text processing for neural machine translation systems, neural architectures for machine translation, the training, and the inference procedures of neural machine translation models.

Chapter 3 reviews the literature of (multi-)domain adaptation in machine translation. We introduce four main sub-problems of (multi-)domain adaptation and provide an overview of the approaches for each sub-problem.

The remainder of the thesis consists of our original work. Chapter 4 proposes a novel multi-criteria evaluation for multi-domain machine translation systems. We reevaluated a large set of model-centric approaches using a relatively large collection of domains. Chapter 5 proposes a (multi-)domain NMT system with cheap computation cost by using a sparse word embedding that nullifies a number of domain-specific units. Chapter 6 proposes several approaches to regularize the residual adapters (Bapna & Firat, 2019b) in the (multi-)domain setting. In this chapter, we propose two variants of the residual adapter that allow us to modularize the domain-agnostic and domain-specific representation and to improve the performance of the adapted NMT model against out-of-domain examples. In Chapter 7, we study multiple dynamic sampling approaches for training the NMT model in the (multi-)domain setting. We propose a novel method that automatically iteratively adapts the sampling distribution to any pre-determined testing distribution. Chapter 8 discusses and reimplements different retrieval-based methods for unknown test domains. We carefully analyze their performance in many domains and demonstrate their strengths and issues in terms of latency, errors.

Finally, in Chapter 9, we draw conclusions in the current state of the development of MDMT. We recall a need for effort to achieve the long-term goal of this approach.

Chapter 2

Neural Machine Translation: a review

In this chapter, we briefly review some basic knowledge of Neural Machine Translation (NMT) which provides the foundation for the experiments of this thesis. Neural Machine Translation was first introduced in 2014 via the work of Cho et al. (2014); Bahdanau et al. (2015). Since then, NMT has been largely developed and outperformed old approaches, including Rule-based Machine Translation (RBMT) and Statistical Machine Translation (SMT) in high-resource languages such as French-English or English-German.

Building a Neural Machine Translation model consists of 3 basic steps, including text tokenization (Section 2.1), training NMT model with pairs of tokenized source and target sentences (Section 2.6) and decoding or translating (Section 2.7). In the first step, each sentence is transformed into a sequence of tokens, which can be words, sub-words, or characters (Section 2.1). The sequence of tokens will be transformed into a sequence of integers. In the second step, given a choice of neural architecture (Sections 2.3, 2.4 or 2.5); the parameters of the NMT model are optimized according to a training objective (Section 2.6). The input of the NMT model during the training consists of a pair of sequences of integers corresponding to a pair of source and target sentences. In the final step, when the NMT model is learned, given any sentence in the source language, the NMT model generates a translation via a decoding algorithm (Section 2.7) such as beam search (Koen, 2004). In the inference step, the input of the NMT model is only the sequence of integers corresponding to the source sentence.

2.1 Text preprocessing for NMT

Text preprocessing includes two steps, including text normalization and text tokenization. Text normalization aims to transform a text into a single canonical form. Text tokenization transforms a sentence into the input format of the NMT model. In practice, text normalization is optional, while text tokenization is obligatory.

The tokenization process consists of transforming a sentence into sequence of symbols called tokens, which will be transformed into sequences of integers and then be served as the input of the NMT model. Text tokenization is an essential step in NMT and needs to

be carefully conducted. We have to tokenize sentences because NMT models only take a sequence of integers as input. In practice, a token can be a word or a part of a word. There are three common types of token: words, sub-words, and characters. These tokens are indexed by a predetermined vocabulary so we can map each token to an integer. The sequence of tokens is converted into a sequence of integers $IDs \in V$ where V is the set of indices of the corresponding vocabulary. The vocabulary of the NMT model is fixed. Any out of vocabulary (OOV) token is mapped to a special token $< UNK >$, which stands for unknown. The size of the vocabulary of an NMT model is chosen to balance the coverage over the processed tokens with a practical constraint on the size of the model. The vocabulary of an NMT model is usually limited to 30-40 thousand types. In the following discussion, we denote Σ_x, Σ_y the source vocabulary and the target vocabulary, respectively. . The tokenization process needs to be reversible. To get the final translation, we convert the sequence of tokens predicted by the NMT model into a normal sentence.

2.1.1 Word tokenization

Word tokenization identifies all unique words in the respective training sets to construct the source and target language vocabularies. Because of the computational constraints, the vocabulary of an NMT model is typically limited to a few tens of thousands of types. The types found in the training sets will be reordered according to their frequency, and the top V most frequent types are selected to form a vocabulary Cho et al. (2014). This tokenization algorithm has the disadvantage that its coverage is relatively small. Consequently, word-based NMT models usually have to scope to out-of-vocabulary tokens Jean et al. (2015); Luong et al. (2015); Li et al. (2016).

2.1.2 Subword tokenization

Subword tokenization is the process of finding an optimal segmentation of words such that a limited set of word-pieces can segment a large vocabulary. The rationale behind the sub-word tokenization is that words are usually composed of several morphemes. For example a plural countable noun is composed of its root and the affix ”s”. By separating the root and the affix, we avoid adding both the singular and the plural form of a noun in our vocabulary and reduce the size of it. In practice, subword tokenization largely increases the coverage of the vocabulary and efficiently handles unseen words. The vocabulary can be built by applying the morphological rules of the language or can be learned by heuristic algorithms such as Byte pair encoding (BPE)(Gage, 1994; Sennrich et al., 2016b). The two more popular sub-word tokenizations are the BPE tokenization (Gage, 1994; Sennrich et al., 2016b) and Sentence-piece tokenization (Kudo, 2018), which are based on 2 different approaches: frequency-based and sampling-based respectively.

BPE tokenization is based on the following algorithm. Given a corpus and an upper bound K of the number of merge operations, BPE tokenization learns a set of at most K merge operations and a set of subwords that allows the formation of any word in that corpus. In principle, words are first segmented into sequences of characters. At each iteration, the BPE algorithm counts the occurrences of each pair of the current types (characters in the beginning), then adds the merge operation of the most frequent pair to its operation set. Next, it redefines the segmentation of every word according to the new operation set and moves to the next iteration. The algorithm stops when it reaches the upper bound K. In the end, frequent words remain unsegmented while rare words become sequences of BPE types. Given a set of BPE operations, BPE tokenization segments a word by first segmenting it into a sequence of characters and then applying merge operations to the characters. BPE operations can be learned jointly from both the source and the target languages, from multiple languages as in multi-lingual NMT or separately from each language. Despite the efficacy in the open-vocabulary NMT, BPE tokenization segments a word into a unique sequence of tokens whereas there exists different segmentation candidates. Because the model only see the sequences of ids, while these sequences encode the same input, NMT handles them as completely different inputs. Therefore, training a NMT model with different segmentation candidates improve the robustness of the model (Kudo, 2018). Provilkov et al. (2020) proposed BPE-dropout which introduces stochastic corruptions in the segmentation procedure of BPE, which leads to producing multiple segmentation candidates within the same fixed BPE framework.

Sentence-piece tokenization also allows many different segmentation candidates for one word but uses a unigram language model to assign a probability to each word segmentation candidate. The motivation of sentence-piece is to enable the NMT model to be trained with multiple segmentation candidates, which will be sampled from a learned distribution over possible candidates. Applying sentence-piece tokenization on the fly allows the NMT model to be robust against the ambiguity raised from the existence of multiple sub-word encoding candidates of a word.

Besides sentence-piece and BPE, there are alternative paradigms for sub-word tokenization such as syllabification (Assylbekov et al., 2017) or linguistically informed tokenization (Ataman et al., 2017; Huck et al., 2017; Machácek et al., 2018).

2.1.3 Character tokenization

Character tokenization segments words into sequences of characters. This tokenization circumvents the problem of finding an optimal sub-word segmentation for multiple languages in multilingual NMT. Furthermore, character tokenization reduces the size of the vocabulary to a small number of written characters. However, the length of the resulting

sequence increases significantly as words are extremely split into character units. As a result computational requirements during training and decoding time increase. First studies on the character-based NMT , including the work of Ling et al. (2015); Luong & Manning (2016), focused on solving the out-of-vocabulary and softmax bottleneck problems associated with word-level models. Costa-jussà & Fonollosa (2016); Chung et al. (2016); Lee et al. (2017); Costa-jussà et al. (2017) proposed variants of character-based model.

2.1.4 Byte-level tokenization

Byte-level tokenization is used to segment the byte-level representation of the text. The rationale behind this tokenization is that byte-level representation could handle character-rich languages such as Japanese and Chinese. However, for the same sentence, the byte-level representation is usually much longer than the character-level representation. Furthermore, taking a sequence of bytes as the input of the NMT model greatly increases the cost. To reduce the length of the input sequence, byte-level tokenization applies BPE tokenization on sequences of bytes. In practice, Wang et al. (2020a) showed comparable performance of byte-level BPE-based NMT compared to BPE-based NMT.

2.2 NMT's main components

In principle, an NMT model consists of 3 parts: 1) a look-up table of word embeddings, 2) an encoder and 3) a decoder. Similar to the SMT approach, an NMT model modelizes the conditional probability of the target sequence given the source sequence, i.e. $P(y|x)$ in which $x = [x_0, \cdots, x_I], y = [y_0, \cdots, y_J]$. Most existing NMT models are auto-regressive, i.e., $P(y|x)$ is factored into a product of a chain of conditional probabilities which predict a target token given the previous predicted target tokens and the source sequence as

$$P(y|x) = \prod_{i=1}^{J} P(y_i|y_{<i}, x). \tag{2.1}$$

We always assume that the target sentence is initialized by a special token named "begin-of-sentence" $< BOS >$, hence, $y_0 =< BOS >$.

The encoder maps the source sequence x to an intermediate representation in a continuous high dimensional vector space. The Decoder takes the representation of the source sequence $Enc(x)$ as input to condition its prediction on the source sequence. At each time step i, the decoder outputs a distribution over the target vocabulary by mapping its i^{th} hidden state to vector space $\mathbb{R}^{|\Sigma_y|}$ where Σ_y is the target vocabulary

$$P(.|y_{<i}, x) = \text{softmax}(Linear(s_i)), \tag{2.2}$$

where *Linear* is a dense layer mapping to $\mathbb{R}^{|\Sigma_y|}$.

The hidden state of the Decoder is computed recursively as

$$s_i = g(s_{i\text{-}1}, y_{i\text{-}1}, c_i), \tag{2.3}$$

using the hidden state of the previous time step, the observation of the previous time step (i.e. the $(i\text{-}1)^{th}$ token) and the context c_i, which is computed from the representation of the source sequence $Enc(x)$ and $s_{i\text{-}1}$.

In order to transform the input sequence of integers into continuous hidden states, Encoder and Decoder have to use a look-up table of word embeddings. A word embedding is a real-valued vector in a high dimension space that represents a token in the vocabulary of the NMT model. The motivation of using word embedding is to transform the input sequence of integers to a sequence of vectors in a continuous space which allows the parameters of the NMT model to be trained with gradient descent-based optimization methods. The lookup table has the size of $|\Sigma_{\{x,y\}}| \times d$ where $|\Sigma_{\{x,y\}}|$ is the size of the corresponding vocabulary, and d is the dimension of word embedding space. Word embeddings are not only used in NMT models but also in Neural language models (Bengio et al., 2003)(NLM). Le et al. (2012); Schwenk (2012) used NLM for phrase-based statistical machine translation. Moreover, word embedding can be trained alone using the Skip-gram model (Mikolov et al., 2013b) or the Continuous Bag of Word model (Mikolov et al., 2013a). After training such models, the resulting word embeddings possess semantic properties so that words having similar meanings or close meanings are mapped to similar vectors in terms of cosine similarity (Collobert et al., 2011; Mikolov et al., 2013b; Collobert & Weston, 2008). The fine-grained semantic representation of word embeddings significantly improves the performance of AI in text classification, text retrieval, etc., and surprisingly enables unsupervised machine translation and unsupervised word translation (Pennington et al., 2014; Levy et al., 2015; Lample et al., 2018; Santos et al., 2020). By using word embeddings, the source sequence is mapped to a sequence of real-valued vectors.

The encoder encodes the source sequence of word embeddings to another sequence of real value vectors (hidden states or contextualized embeddings) (Cho et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017), in a high dimension space called a latent space. This process aims to mix the representation of each token with ones of the context surrounding that token. The context of a word is the set of words surrounding that word. Combining the representation of a word with its context allows the NMT model to condition the translation of that word on its context. The encoder can combine the state of the token with one of its preceding tokens in a Recurrent encoder, with ones of the sur-

rounding window in a Convolutional encoder, or with ones of the whole sentence in an Attention-based encoder. We illustrate the range of context captured by those three encoders in figure 2.1. Each encoding paradigm has its advantages and disadvantages. The Recurrent encoder respects the order of tokens because it consumes tokens one by one from left to right. However, it is very slow to encode the input sequence. Convolutional encoder and Attention-based encoder encode all input tokens simultaneously that brings a great advantage in speed. But allowing direct connections between states prevents Convolutional encoders and Attention-based encoders from apprehending the sequence's order. Therefore they have to use positional embedding to know the position of each token.

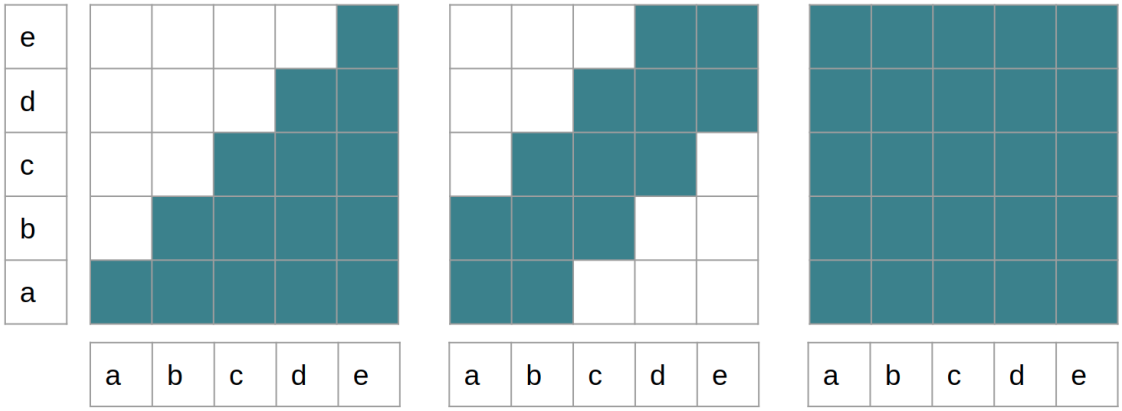


Figure 2.1: Illustration of context range at each token in different encoding mechanism. From left to right: Recurrent encoder, Convolutional encoder, Attention-based encoder. The example sequence is $[a, b, c, d, e]$ and each colored column represent the context range of the corresponding token.

The decoder works similarly to a language model as it predicts one token per time step. However, the decoder conditions its prediction on the source sequence. Therefore, the decoder takes the output of the encoder as its inputs. An Auto-regressive decoder conditions its prediction on the predictions of previous steps and the source sequence. Because all of our experiments use auto-regressive NMT, from now, a decoder is an auto-regressive decoder if there is no other specification. The decoder usually uses the same neural architecture as the encoder. However, unlike the encoder, the range of context of a token is strictly limited to its preceding tokens. Because the hidden state of the decoder is computed from the previous hidden states and the observation of the previous step, we need to initialize the 0^{th} hidden state s_0 (optional) and the 0^{th} token. That is why we always begin the target sequence by the token $< BOS >$, and the decoder starts predicting from the second token on. For example, if $[a, b, c, d, e]$ is predicted by the decoder, the prediction of token a is conditioned by source sequence x and $< BOS >$; the prediction of token b is conditioned by source sequence x and $[< BOS >, a]$ and so

on. Besides, the decoder needs a signal to stop its generative prediction. We always end a prediction by "end-of-sentence" token or $< EOS >$. Therefore, instead of predicting $[a,b,c,d,e]$, the decoder predicts $[a,b,c,d,e,< EOS >]$. Concerning the construction of hidden states, the Recurrent decoder usually initializes s_0 by the last hidden state of the encoder followed by a linear transformation. In contrast, the Convolutional decoder and the Attention-based decoder do not need to initialize s_0 as every hidden state directly accesses the predictions preceding its time step without going through its preceding state. We illustrate the difference between decoding paradigms in the figure 2.2.

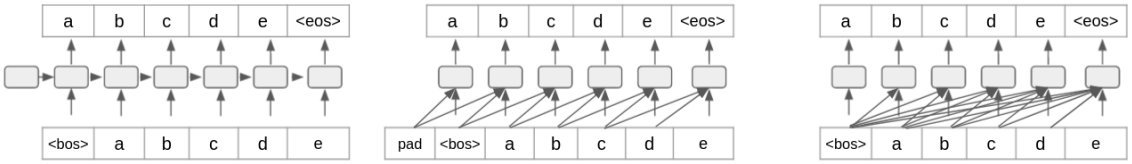


Figure 2.2: From left to right: Recurrent decoder, Convolutional decoder, Attention-based decoder. The example sequence is $[a,b,c,d,e]$. The figure illustrates only one layer of the decoder.

NMT's architectures are usually a stack of multiple layers. As described above, the input source sequence is mapped to a sequence of word embeddings. This is considered the 0^{th} layer of the Encoder. The i^{th} layer is built upon the $(i\text{-}1)^{th}$ layer by applying the same encoding mechanism, which can be recurrent layer, convolutional layer or self-attention layer, to the output of the $(i\text{-}1)^{th}$. We illustrate different multi-layer decoders in figure 2.3. For example, Vaswani et al. (2017) stacked 6 Transformer layers in both the encoder and the decoder of their NMT model. Deep NMT models are able to learn from very large-scale of parallel data (Ott et al., 2018) and continually create new state-of-the-art performances. However, deep NMT models are harder to train because the gradient flow has to back-propagate through many layers. In order to prevent the gradient flow from vanishing, which happens when the value of the output of the linear transformation in some layer jumps outside the domain of the activation function, (He et al., 2016) proposes using residual connections, which replaces $f(x)$ by $f(x)+x$ where x is the output of the lower layer and $f(.)$ is the transformation of the layer, to transit from the lower layers to their following layers. By using residual connections, a fraction of the gradient still reaches the lower layer and continues to propagate until the lowest layer.

Deep NMT models also suffer from Internal Covariate Shift in which the distribution of the value of each layer significantly changes due to the change of the parameters of the models. In deep neural network, the distribution of the value of high layers is highly affected by the parameters of the lower layers and can be dramatically shifted by a small change in the value of those parameters. Large shift can push the value of the layer to the saturation zone of activation function where the gradient is extremely small. In

practice, the saturation problem can be mitigated by using the Rectified Linear Units $RELU(x) = max(x, 0)$ (Nair & Hinton, 2010). Recently, (Ioffe & Szegedy, 2015; Ba et al., 2016) propose different normalization methods to stabilize the value of layers so that they are not easily pushed to saturation zone of activation function. In principle, Normalization methods re-scale and re-center the distribution of the value of each layer with learnable mean and learnable variance. Normalization methods prove to be very helpful in practice. For example, Layer normalization must be included in every layer of Attention-based NMT (Vaswani et al., 2017).

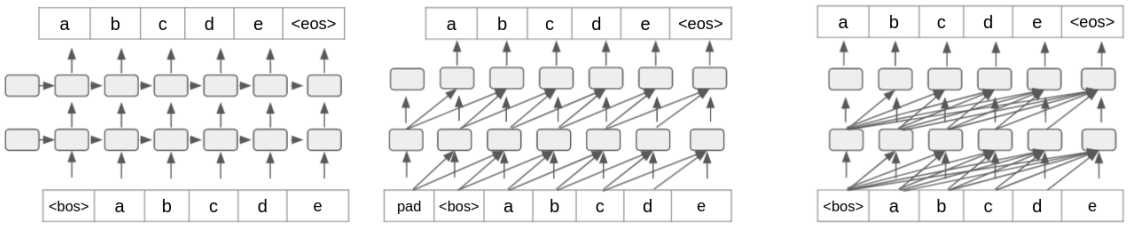


Figure 2.3: From left to right: Recurrent decoder, Convolutional decoder, Attention-based decoder. The example sequence is $[a, b, c, d, e]$. The figure illustrates only two layers of the decoder.

2.3 Recurrent neural machine translation

This section reviews the very first NMT architecture, the Recurrent neural machine translation architecture (RNMT). RNMT is composed of a Recurrent encoder, a Recurrent decoder, and tables of word embeddings. The Recurrent encoder and the Recurrent decoder usually use the same type of Recurrent neural network (RNN) layers, such as Gated recurrent unit (GRU) and Long-short term memory (LSTM), which we will explain in the following section. RNMT is strictly auto-regressive as each hidden state in the encoder/decoder has to go through every intermediate state to assess the information of any time step before it. The hidden states of RNMT inherit the ordering information, which is an advantage over Convolutional neural machine translation (CNMT) and Attention-based neural machine translation (ANMT). However, a lack of straightforward connections between the positions of the input sequence causes many difficulties in the training of RNMT, for example the vanishing gradient problem in backpropagation through time Pascanu et al. (2013).

2.3.1 GRU, LSTM layers

Gated recurrent units (GRU) and Long-short term memory units (LSTM) are the two most popular layers in the group of Recurrent neural networks. They follow the auto-regressive

paradigm by constructing the hidden states one by one as follows

$$h_t^l = f(h_t^{l\text{-}1}, h_{t\text{-}1}^l) \tag{2.4}$$

where $h_t^{l\text{-}1}$ is the hidden state at time step t of the $(l\text{-}1)^{th}$ layer, the 0^{th} layer is the sequence of word embeddings; the mapping f can be GRU cell or LSTM cell, which will be explained below.

LSTMs were first introduced by Hochreiter & Schmidhuber (1997). They use 4 gating functions including input gate i, output gate o, forget gate f and memory cell c. At each time step t, the contextualized embedding h_t is computed as follows

$$\begin{aligned} f_t &= \sigma_g(W_f h_t^{l\text{-}1} + U_f h_{t\text{-}1} + b_f), \\ i_t &= \sigma_g(W_i h_t^{l\text{-}1} + U_i h_{t\text{-}1} + b_i), \\ o_t &= \sigma_g(W_o h_t^{l\text{-}1} + U_o h_{t\text{-}1} + b_o), \\ \tilde{c}_t &= \sigma_c(W_c h_t^{l\text{-}1} + U_c h_{t\text{-}1} + b_c), \\ c_t &= f_t \odot c_{t\text{-}1} + i_t \odot \tilde{c}_t, \\ h_t &= o_t \odot \sigma_h(c_t), \end{aligned} \tag{2.5}$$

where σ_g is the sigmoid function, σ_c is the hyperbolic tangent function, σ_h is either the hyperbolic tangent function or the identity function and $\odot$ is the element-wise multiplication. These functions are applied element-wise to intermediate vectors in the equations.

The motivation behind this highly complex structure is to stabilize the exploding/diminishing gradient flow (Pascanu et al., 2013) induced by back-propagation through time (BPTT) (Hochreiter & Schmidhuber, 1997). The second architecture GRU, which was proposed by Cho et al. (2014), mitigates the complexity of LSTM by using only three gates as follows.

$$\begin{aligned} z_t &= \sigma_g(W_z h_t^{l\text{-}1} + U_z h_{t\text{-}1} + b_z) \\ r_t &= \sigma_g(W_r h_t^{l\text{-}1} + U_r h_{t\text{-}1} + b_r) \\ \hat{h}_t &= \sigma_h(W_h h_t^{l\text{-}1} + U_h(r_t \odot h_{t\text{-}1}) + b_h) \\ h_t &= (1 - z_t) \odot h_{t\text{-}1} + z_t \odot \hat{h}_t \end{aligned} \tag{2.6}$$

Where σ_h is a hyperbolic tangent function while other notations are the same as in the equations 2.5.

2.3.2 RNN encoders

RNN encoders use LSTM or GRU layers to encode the source sequence. RNN encoders can use more than one layer to capture more fine-grained language representations (Li et al., 2020). The 0^{th} layer is a sequence of word embeddings, which are extracted from the look-up table of the source side using the word ordering provided by the source se-

quence.

2.3.2.1 Bidirectional RNN encoders

Unlike the decoder, the encoder is not constrained to process the input sequence from left to right. Effectively, the context of one token in the source sequence contains not only its preceding neighbors but also its following neighbors. Therefore, encoding the source sequence from left to right is not enough to fully describe the context of each token. To increase the coverage of contextualized embedding, the encoder process the source sequence both from left to right and from right to left at the same time. Such encoders are deemed bidirectional. Bidirectional encoding results in two sequences of contextualized embeddings; the encoder simply combines two contextualized embeddings of a token into one real vector via either concatenation or summation. The resulting contextualized embedding improves the representation of each word with information regarding its neighbours on the right and on the left.

2.3.3 RNN decoders

RNN decoders predict the target sequence from left to right, one token per time step. It initializes the 0^{th} hidden state by zero vector or a linear transformation of the last hidden state of the last layer of the encoder. The following section will discuss on an important component of the NMT model, which are attention mechanisms. As the hidden representation of the decoder at each step is computed as follows

$$s_i = g(s_{i\text{-}1}, y_{i\text{-}1}, c_i), \tag{2.7}$$

where c_i is the context vector. c_i is computed via an attentional mechanism using s_i and $Enc(x)$, which will be explained in Section 2.3.3.1.

The prediction probability will be computed as follows

$$p(y_i|s_i, y_{i\text{-}1}, c_i) = \text{softmax}(Dense(t_i))_{y_i} \tag{2.8}$$

where y_i is an index of the target vocabulary, *Dense* is a dense layer, whose output is of dimension $|\Sigma_y|$, and t_i is computed as follows

$$\begin{aligned} t_i &= \left[max\left\{\tilde{t}_{i,2j\text{-}1}, \tilde{t}_{i,2j}\right\}\right]_{j=1,\cdots,d}^{T}, \\ \tilde{t}_i &= U_0 s_{i\text{-}1} + V_0 Emb(y_{i_1}) + C_0 c_i, \end{aligned} \tag{2.9}$$

where $Emb(y_{i\text{-}1})$ is the word embedding of the token $y_{i\text{-}1}$, $U_0 \in \mathbb{R}^{2l \times d}$, $V_0 \in \mathbb{R}^{2l \times d'}$, and

$C_0 \in \mathbb{R}^{2l \times d}$ for a uni-directional encoder and $C_0 \in \mathbb{R}^{2l \times 2d}$ for a bi-directional encoder.

2.3.3.1 Attention mechanisms

An attentional mechanism consists of 3 components: Query vectors, Key vectors, and Value vectors. Given a sequence Q_i, $i \in [1 \cdots n]$, K_j, $j \in [1 \cdots m]$ and V_j, $j \in [1 \cdots m]$, the results of the attentional mechanism composed by those vectors will be as follows

$$\text{Attention}(Q,V,K)_i = \sum_{j=1}^{m} \frac{exp(sim(Q_i,K_j))}{\sum_{p=1}^{m} exp(sim(Q_i,K_p))} * V_j, i \in [1,\cdots,m], \tag{2.10}$$

where the function $sim(x,y)$ can be the standard dot product $< x,y >$ (Vaswani et al., 2017), a generalized dot product $< x,W_a * y >$ or $< v_a, tanh(W_a * [x,y]) >$ (Luong & Manning, 2015; Bahdanau et al., 2015).

The attention mechanism manages and quantifies the dependences between the input sequence and the output sequence (e.g., source contextualized embeddings and target contextualized embeddings), or the input sequence itself (e.g., self-attention layers in Transformer (Vaswani et al., 2017)). In the RNN MT model, the attentional mechanism is used to capture the dependences of each token in the target sequence on the tokens in the source sequence. For example, Bahdanau et al. (2015) computed a context vector at i^{th} time step in the decoder as follows

$$\begin{aligned} c_i &= \textstyle\sum_j \alpha_{ij} h_j, \\ \alpha_{ij} &= \frac{exp(e_{ij})}{\sum_k exp(e_{ik})}, \\ e_{ik} &= sim(s_{i\text{-}1}, h_k), \end{aligned} \tag{2.11}$$

where h_j is the output of the last layer of the encoder, $s_{i\text{-}1}$ is the hidden state at the $(i\text{-}1)^{th}$ time step of the last layer of the decoder.

2.4 Convolutional neural machine translation

The convolutional neural network was successfully applied to the MT task in the work of Gehring et al. (2017) that outperformed the current state-of-the-art performance of the RNMT. As we mentioned in the previous section, convolutional neural machine translation (CNMT) does not construct the hidden states iteratively in one direction. The model considers the sequence as an image, of which each column of pixels is a word embedding, and applies convolutional kernels on it. Therefore, CNMT is much faster than RNMT. We give the detail of the convolutional encoder and the convolutional decoder in the following sections.

2.4.1 Convolutional encoders

Concretely, each layer of a convolutional encoder contains a one dimensional convolution kernel followed by a non-linear activation function. We denote h_i^l the i^{th} hidden state of the l^{th} layer. Those hidden states are computed as follows

$$h_i^l = v\left(W^l\left[h_{i-\frac{k}{2}}^{l-1}, \cdots, h_{i+\frac{k}{2}}^{l-1}\right] + b_w\right) + h_i^{l-1}, \tag{2.12}$$

where $W^l \in \mathbb{R}^{2d \times kd}$, $b_w \in \mathbb{R}^{2d}$, d is the dimension of hidden states as well of word embeddings, k is the width of the kernel, the activation function v is Gated Linear Unit (Gehring et al., 2017) defined as follows

$$v([A,B]) = A \odot \sigma(B), \tag{2.13}$$

where $\odot$ is the element-wise multiplication, σ is the sigmoid function.

2.4.2 Convolutional decoders

Unlike the convolutional encoder, in which each hidden states has access to its left and right neighbors, the decoder only allows left accesses to avoid conditioning the predictions on the following tokens, which remain unknown before the prediction during the inference. Therefore, Gehring et al. (2017) appended k-1 padding tokens in the left side of the output sequence, e.g PAD, PAD,$< BOS >$,je,t',$aime$ for convolution kernel of size 3 so that $\left[PAD, PAD, < BOS >\right]$ predicts je, $\left[PAD, < BOS >, je\right]$ predicts t' and so forth. The convolutional decoder also uses an attention mechanism to improve the performance for long sentences. Gehring et al. (2017) proposed a slightly different version from ones of Luong & Manning (2015); Bahdanau et al. (2015). For each l^{th} decoder layer, the query will be a combination of the hidden state h_i^l and the word embedding of the previous token g_i as follows

$$Q_i^l = W_d^l h_i^l + b_d^l + g_i. \tag{2.14}$$

The keys are still the hidden states of the last layer of the encoder z_j^u. The values are the combinations of the hidden states z_j^u and the word embedding e_j as follows

$$V_j^l = z_j^u + e_j. \tag{2.15}$$

The attention score is the dot product between the query vector and the key vector followed by the softmax function as follows

$$\alpha_{ij} = \frac{exp(Q_i^l \cdot z_j^u)}{\sum_{t=1}^{m} exp(Q_i^l \cdot z_t^u)}. \tag{2.16}$$

The context vector c_i^l will be as follow

$$c_i^l = \sum_{j=1} \alpha_{ij} V_j^l \tag{2.17}$$

Once c_i^l has been computed, it is simply added to the output of the corresponding decoder layer h_i^l.

2.4.3 Positional embeddings

Gehring et al. (2017) proposed using embeddings corresponding to each position of the input sequence. The purpose is to equip the CNMT model with a sense of order as the convolution kernel does not take into account the order of tokens in the input sequence. Effectively, if we interchange the position of tokens outside the window of the kernel, the value of the hidden state does not change. Positional embeddings are real value vectors having the same dimension as word embeddings. Positional embeddings are added to word embeddings of the corresponding position before passing to the first layer.

2.5 Attention-based neural machine translation

The Transformer architecture was first introduced by Vaswani et al. (2017) and has quickly become the state-of-the-art architecture not only in MT but also in language modeling (LM) (Devlin et al., 2019; Conneau & Lample, 2019a; Brown et al., 2020), text summarization (Zhang et al., 2020) etc. The Transformer model's power relies on the attentional mechanism, which was discussed in the previous section 2.3.3.1. The Transformer model consists of a fully attention-based encoder and decoder.

2.5.1 Transformer encoders

The Transformer encoder consists of layers made of a multi-head self-attention sub-layer followed by a position-wise fully connected feed-forward network. The multi-head self-attention sub-layer is an extension of the self-attention sub-layer and is described by the following equation

$$\begin{aligned} MultiheadAttention\big(Q,V,K\big) &= Concat\big[head_0,\cdots,head_h\big]W_0 \\ head_i &= Attention\big(QW_i^Q, VW_i^V, KW_i^K\big), \end{aligned} \tag{2.18}$$

where $W_i^Q, W_i^V, W_i^K \in \mathbb{R}^{d_k \times d_h}$ with $d_h \times h = d_k$, d_k is the dimension of word embedding space and also the size of the Transformer model. Unlike the version (2.21) in Section

2.3.3.1, the attentional mechanism is simply as follows,

$$Attention\big(Q,V,K\big) = \text{softmax}\,\big(\frac{QK^T}{\sqrt{d_k}}\big)V. \tag{2.19}$$

The feed-forward network is designed as follows

$$FFN(x) = ReLu(xW_1 + b_1)W_2 + b_2, \tag{2.20}$$

where $W_1 \in \mathbb{R}^{d_k \times d_b}$,$W_2 \in \mathbb{R}^{d_b \times d_k}$,$b_1 \in \mathbb{R}^{d_b}$,$b_2 \in \mathbb{R}^{d_k}$. The final detail is that the output of each sub-layer has to pass through a Layer-Normalization sub-layer (Ba et al., 2016). In conclusion, the contextualized embedding of the l^{th} layer of the Transformer encoder will be as follows

$$\begin{aligned} \tilde{h}^l &= LN\Big(Multihead\big(h^{l\text{-}1},h^{l\text{-}1},h^{l\text{-}1}\big) + h^{l\text{-}1}\Big), \\ h^l &= LN\Big(FFN\big(\tilde{h}^l\big) + \tilde{h}^l\Big), \end{aligned} \tag{2.21}$$

where *LN* is a Layer-Normalization sub-layer.

2.5.2 Transformer decoders

The Transformer decoder consists of layers made of a multi-head self-attention sub-layer followed by a multi-head cross-attention sub-layer then by a position-wise fully connected feed-forward network. The multi-head self-attention sub-layer and the feed-forward network have the same design as those in the Transformer encoder. The multi-head cross-attention sub-layer of the l^{th} layer of the decoder uses the output of the last layer of the encoder as keys and values, the output of the l^{th} self-attention sub-layer as queries

$$\begin{aligned} \tilde{s}^l &= LN\Big(Multihead\big(s^{l\text{-}1},s^{l\text{-}1},s^{l\text{-}1}\big) + s^{l\text{-}1}\Big), \\ \bar{s}^l &= LN\Big(Multihead\big(\tilde{s}^l,h^u,h^u\big) + \tilde{s}^l\Big), \\ s^l &= LN\Big(FFN\big(\bar{s}^l\big) + \bar{s}^l\Big), \end{aligned} \tag{2.22}$$

where s^l,$s^{l\text{-}1}$ are the outputs of the l^{th} and $(l\text{-}1)^{th}$ layers of the decoder respectively, h^u is the output of the last layer of the encoder.

In order to prevent the future information in the decoder, at each time step i^{th}, the attention scores of tokens at positions after i^{th} are masked by zero.

2.5.3 Positional embeddings

Similar to the convolutional encoder/decoder, the Transformer encoder/decoder does not respect the order of tokens as it fully connects every pair of tokens in parallel. In order to represent the position of tokens in the sequence, Vaswani et al. (2017) proposed the use of positional embeddings. Unlike Gehring et al. (2017)'s positional embedding, this version is not parameterized as given the size of word embedding d_k , the positional embedding of position i^{th} is defined as

$$\begin{aligned} PE\big(pos, 2i\big) &= sin\big(\frac{pos}{1000^{\frac{2i}{d_k}}}\big) \\ PE\big(pos, 2i+1\big) &= cos\big(\frac{pos}{1000^{\frac{2i}{d_k}}}\big). \end{aligned} \tag{2.23}$$

The positional embedding will be added to the corresponding word embeddings of the i^{th} token in the input sequence.

2.6 Training NMT models

The purpose of training NMT models is to find optimal values for their parameters so that the error of the model is minimized in inference over unseen test sets. To learn these optimal values, we need 3 type data sets including training set, validation set, and testing set. The training set is used to optimize the model's parameters via statistical learning algorithms such as Maximum likelihood estimation (MLE)(Baum & Wilczek, 1987). The testing set is used to evaluate the model once it's optimized. The performance on the testing set shows us how good the model is and is used to compare different models. The validation set is not used to learn the model's parameters nor to evaluate the model but to prevent the "over-fitting" of the optimization. Effectively, an NMT model can be trained until very small error in the training set but this will result in poor generalization on the test set. A common practice is to perform early stopping. During the training, the model is evaluated on the validation set for every K iterations. The learning is represented by 2 learning curves, including the error on training set and the error on the validation set. Training stops when the validation error does not improve after a predetermined number of consecutive evaluations.

Concerning the optimization of the model on the training set, we usually use MLE, i.e.

$$\hat{\theta} = \underset{\theta}{argmax}\, \mathbb{E}_{x\sim\mathscr{D}_e, y\sim g(x)} logP(y|x;\theta), \tag{2.24}$$

where $\mathscr{D}_e$ is the train distribution over the set of sentences, Ω_e in language e and g is a hypothetical translation function, $g : \Omega_e \rightarrow \Omega_f$, which needs to be learned by our model

(Ω_f is set of sentences in the language f). According to Equation 2.1, MLE is equivalent to minimizing the cross-entropy loss

$$\begin{aligned} L_{CE}(\theta, \mathscr{D}_e, g) &= -\mathbb{E}_{x\sim\mathscr{D}_e, y\sim g(x)} \sum_{i}^{l_y} logP(y_i|y_{<i}, x; \theta), \\ \hat{\theta} &= \underset{\theta}{argmin}\, L_{CE}(\theta, \mathscr{D}_e, g). \end{aligned} \tag{2.25}$$

In order to optimize this function, we often use the gradient descent method, which is one of the oldest approaches for optimizing continuous functions (Cauchy, 1847). The gradient is computed by the back-propagation algorithm (Rumelhart et al., 1988). Like many deep learning models, the NMT model is usually trained with a massive amount of data that makes gradient descent computationally infeasible. Therefore, stochastic gradient descent (SGD) is proposed to mitigate the computational burden of large-scale models (Robbins & Monro, 1951; Kiefer & Wolfowitz, 1952; Bottou, 2010). In theory the loss is (2.24), so we approximate the gradient with a small batch. SGD samples a batch of examples from training set, calculates the gradient of the loss over this batch, then updates the parameters according to this gradient.

2.6.1 Tips and tricks in training an NMT model

Deep Neural networks are usually very hard to train. Effectively, back-propagation through time in RNMT usually creates exploding or vanishing gradients (Glorot & Bengio, 2010; Pascanu et al., 2013). Gradient clipping (Pascanu et al., 2013), Truncated back-propagation (Jaeger, 2002) are proposed to mitigate these problems.

Large NMT models are easily over-fitted to training data. Srivastava et al. (2014) proposed randomly zeroing-out a subset of parameters during one training iteration, which prevents the whole model from being fitted to one example. The method is called dropout. We could interpret dropout as an ensemble method that allows training many sub-networks in one training and ensembles them in testing. In practice, dropout is essential to train neural models in general.

Besides, log-likelihood maximization (2.24) assumes that a ground-truth label is far more likely than all other labels, excessively discriminates between the likelihood of training examples and the likelihood of the ones that do not appear during training. The log-likelihood maximization can result in over-fitting to the training data, reducing the model's generalization in testing. To mitigate this problem, we adopted the 'label smoothing' technique from computer vision Szegedy et al. (2016). The technique replaces the loss (2.25) by a smooth approximation defined as follows

$$
\begin{aligned}
L_{CE}(\theta, \mathscr{D}_e, g) &= -\mathbb{E}_{x\sim\mathscr{D}_e, y\sim g(x)}\Bigg[\sum_{i}^{l_y}\big((1-\varepsilon) * logP(y_i|y_{<i}, x; \theta) \\
&+ \frac{\varepsilon}{|\Sigma_y|} * \sum_{y'\in\Sigma_y} logP(y = y'|y_{<i}, x; \theta)\big)\Bigg],
\end{aligned}
\tag{2.26}
$$

2.7 Inference with an NMT model

An NMT model translates a source sentence x by searching the target sequence y that gives the highest probability conditioned on x,

$$\hat{y} = \underset{y}{\text{argmax}} P(y|x; \theta) \tag{2.27}$$

However, the search space of y is of infinite dimension, making exact search intractable. Beam search (Och & Weber, 1998) is the most common inference algorithm in Neural Machine Translation and Statistical Machine Translation. For autoregressive NMT models, a single output token is produced at each inference step j. The prediction at step j is conditioned by x and the partial translation hypothesis up to step j

$$\hat{y}_j = \underset{y_j\in\Sigma_y}{\text{argmax}} P(y_j|y_{<i}, x, \theta). \tag{2.28}$$

Beam search tracks K most probable translation hypotheses. Beam search starts with K empty hypotheses, which are initialized by the "begin of sentence" token $< BOS >$. In the j^{th} inference step, for the n^{th} partial hypothesis $[y^n_{<j}]$, the top-K most probable tokens according to $P(.|y_{<j}, x; \theta)$ are picked and appended to the current hypothesis

$$\hat{y}^n_j \in \underset{y_j\in\Sigma_y}{Top_K} P(y_j|y_{<j}, x, \theta). \tag{2.29}$$

The search space, therefore, is extended to $K * K$ hypotheses. Beam search selects only the top K hypotheses from these $K * K$ hypotheses. It stops extending an hypothesis when $< EOS >$ is predicted or the hypothesis reaches the predefined length limit.

Beside the left-to-right decoding, there are several variant decoding algorithms including non-monotonic decoding (Welleck et al., 2019), non auto-regressive decoding (Gu et al., 2017), and synchronous bidirectional decoding (Zhou et al., 2019). Because the decoding algorithms are not central in this thesis, we would like to restrict ourselves to this brief description.

2.8 MT evaluation

The evaluation of MT systems can be done automatically by comparing n-grams of generated translations and n-grams of gold references. The most popular MT metric is BLEU (Papineni et al., 2002). Recently, Post (2018) proposed standardizing hypotheses of MT systems before calculating the BLEU score.

BLEU is computed at the corpus-level, i.e., it compares a corpus of hypotheses and the corpus of references. BLEU score is the geometric average of n-gram precisions, including 1-gram, 2-grams, 3-grams and 4-grams weighted by brevity penalty (BP)

$$BLEU = BP \times exp(\frac{1}{4}\sum_{i=1}^{4} logp_i). \tag{2.30}$$

The n-gram precision p_n is computed as follows

$$p_n = \frac{\sum_{hyp \in hyps} \sum_{n\text{-}gram \in n\text{-}grams} min(Count(Ref, n\text{-}gram), Count(hyp, n\text{-}gram))}{\sum_{hyp \in hyps} \sum_{n\text{-}gram \in n\text{-}grams} Count(Ref, n\text{-}gram)}, \tag{2.31}$$

Where $Count(C, g)$ is the number of occurrences of the n-gram g in the corpus C.

The Brevity penalty computed as follows

$$BP = \begin{cases} 1 & \text{if } |c| > |r| \\ exp(1 - \frac{|r|}{|c|}) & \text{otherwise.} \end{cases} \tag{2.32}$$

Where c is the total length of the hypothesis corpus, r is the total length of the reference corpus. The Brevity penalty assures that a high-scoring candidate translation must also match the reference translations in length.

Chapter 3

(Multi-)domain adaptation in neural machine translation

The domain mismatch problem is one of the main challenges in machine translation (Koehn & Knowles, 2017) in which the test distribution (the target domain) is different from the train distribution (the source domain). Domain adaptation approaches address the problem with one target domain, while multi-domain adaptation aims to improve the performance of an NMT model in multiple domains. While the problem has been actively studied for a long time, MT (multi-)domain adaptation literature lacks a unified foundation. Several previous works aimed to disambiguate the notion of a domain in MT domain adaptation, such as van der Wees et al. (2015); van der Wees (2017); Saunders (2021). The thesis of Saunders (2021) gave us a good review of domain adaptation. Chu & Wang (2018a) regrouped domain adaptation approaches into two main categories, including the data-centric group and the model-centric group but also forgot to mention their use case. Because none of the reviewed methods in Chu & Wang (2018a) solved all the situations of the domain mismatch problem, it is essential to summarize which method solves which case. Furthermore, these works only provided a facet of multi-domain adaptation. In this chapter, we want to propose a complete overview of this problem. First, we point out four main cases in (multi-)domain adaptation. Then we use a unique mathematical framework to describe these situations. Finally, we match domain adaptation methods to each case creating a Cartesian coupling of a technique and its application case.

We divide this chapter into five sections. In Section 3.1, we would like to discuss how we usually define a domain, how translations differ between domains, and the importance of domain adaptation in real applications. In Section 3.2 we would like to regroup two notions, domain adaptation and multi-domain adaptation, then divide the problem MT (multi-)domain adaptation into four main sub-problems. We dedicate the four following sections to these sub-problems. In each of these sections, we review groups of approaches (data-centric or model-centric), according to Chu & Wang (2018b), that match the requirement of the corresponding problem.

3.1 What is a domain?

In a classical machine learning context such as the binary classification problem, Ben-David et al. (2010a) defined a domain by a pair of a distribution $\mathscr{D}_x$ on the input space Ω_x and a labeling function $g : \Omega_x \to \{0,1\}$. In machine translation context, the labeling function will be $g : \Omega_e \to \Omega_f$ where Ω_e and Ω_f are the set of sentences of the language e and the language f respectively. Denote 2 different domains, $\left(\mathscr{D}_e^S, g^S\right)$ and $\left(\mathscr{D}_e^T, g^T\right)$. Domain adaptation is required when we train a machine learning model (statistical or neural) with the data generated by $\left(\mathscr{D}_e^S, g^S\right)$ but apply it to the data generated by $\left(\mathscr{D}_e^T, g^T\right)$. In principle, there is no guarantee that the model performs well in the second domain. However, we could aim to exploit some sharing knowledge between the two domains; for example, Blitzer et al. (2006); Miller (2019) explored pivot features, which are features that frequently occur in the two domains and similarly contribute to the predictions in both domains. This paradigm was applied to multi-domain machine translation in the work of Britz et al. (2017). However, Pham et al. (2021) reevaluated the method with a wider range of domains and observed a low performance.

In the machine translation literature, "a domain is defined by a corpus from a specific source" (Koehn & Knowles, 2017). van der Wees et al. (2015); van der Wees (2017) identified the following elements of a text which influence the translation the most.

- Topic: the subject of the text such as medical, news, IT, or religious. A topic owns its particular vocabulary. These items can not be transferred between distant topics such as medical and religion.
- Genre: the purpose of the text such as education, talk, report, or instruction. It identifies groups of texts that share a common form of transmission, purpose, and discourse properties. We characterize genre by textual style, the structure of the text, etc.

Carpuat et al. (2013) provided an essential insight into domain adaptation. The authors carefully analyzed the sources of error when translating in a new domain. They identified unseen words and senses as the primary sources of error. According to Carpuat et al. (2013), there are four kinds of mistakes that an SMT system makes when translating in a new domain:

- SEEN: unseen words in the new domain,
- SENSE: unseen domain-specific translations for known words,
- SCORE: wrong preference for non-new-domain translations,
- SEARCH: search algorithm chooses wrong words.

These errors are detected by analyzing the word alignment component of an SMT model. However, NMT models do not use word alignments explicitly in their mechanism. More-

over, the attention mechanisms in NMT architectures are usually hypothesized to play the same role as word alignment components implicitly. Consequently, SEEN, SENSE, SCORE, and SEARCH errors have not been used to analyze the performance of an NMT model in an unseen domain.

Despite many specifications, domains share the same grammar of the corresponding language and a relatively large vocabulary. Furthermore, domains from similar topics such as medical and scientific can share many common domain-specific words. Domains from a similar genre, such as administrative, can share the same formality. Therefore, the data of one domain can be helpful to improve the performance of a NMT model in other domains. The problem of (multi-)domain adaptation is to best exploit the similarity between domains and the specialization for each domain. These two goals are contrary to each other but both essential for a (multi-)domain MT system.

Solving the MT (multi-)domain adaptation problem is essential for deploying MT in a real context. Machine Translation has applications in many sectors, such as translating legal documents, news, scientific documents, books, movie subtitles, etc. Every domain has its specific vocabulary, registers (formal or informal), and genres (e.g., talk, instruction). Therefore, tailoring MT models to a target domain is essential to achieve good translation in that domain. In practice, MT models (SMT, NMT) trained with domain-related data always perform much better in the domain of interest than ones trained with the same amount of less relevant data (Sennrich, 2013; Saunders, 2021). The more domain-relevant data is available, the better the MT system performs in the target domain. However, not every domain has enough data to train an MT model. Moreover, state-of-the-art models such as Transformer models will need millions of parallel sentence pairs to learn its parameters. Therefore, we have to work around the situation where there is very little or no data. Domain Adaptation aims to improve the performance of an MT model in low-resourced domains. Multi-domain adaptation seeks to achieve the best performance in multiple domains at once. The domains of interest in multi-domain adaptation are not limited to be low-resourced domains. The motivation of having one model adapted to many domains is to optimize the storage, the training time, and deployment time. Having one model per domain increases the storage, the time retrieving a model, and thus the translation latency. Online translation services, such as Google Translate, Systran Translate, or DeepL Translate, have to translate text from any possible domain while minimizing the latency of translation to be beneficial. In conclusion, the variety of text between application fields requires domain adaptation, while fast and robust translation requires multi-domain adaptation.

3.2 Machine translation (multi-)domain adaptation - a multi-faceted problem

3.2.1 From domain adaptation MT to multi-domain adaptation MT

Domain adaptation and multi-domain adaptation do not have same motivation as one focuses on low-resourced domain whereas the other focuses on adapting to as many as possible domains, they can however cast under one general framework. Formally, train instances are distributed according to a mixture $\mathscr{D}_e^S$ such that $\mathscr{D}_e^S(x) = \sum_{d=1}^{n_d} \lambda^s(d)\mathscr{D}_e^d(x)$, with $\{\lambda^s(d), d = 1 \dots n_d\}$ the mixture weights satisfying $\sum_d \lambda^s(d) = 1$. The target domains are represented in the test distribution which is also a mixture of $\mathscr{D}_e^T(x) = \sum_{d=1}^{n_d} \lambda^t(d)\mathscr{D}_e^d(x)$, with $\{\lambda^t(d), d = 1 \dots n_d\}$ the mixture weights satisfying $\sum_d \lambda^t(d) = 1$. Domain adaptation solves the case where λ^t is an one-hot vector while multi-domain adaptation happens to solve the case where λ^t is not an one-hot vector. We illustrate this formulation in figure 3.1.

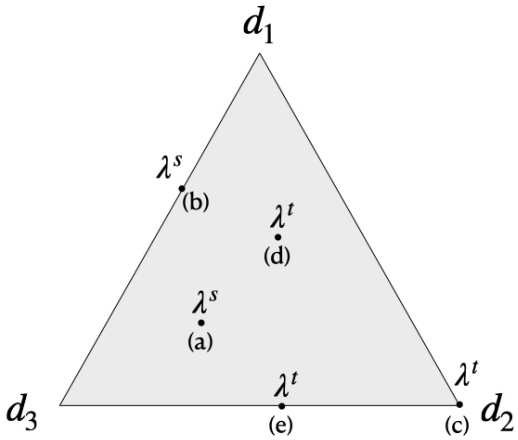


Figure 3.1: Training and testing with distribution mismatch. We consider just three domains, and represent vectors of mixture weights $\boldsymbol{\lambda}^s$ and $\boldsymbol{\lambda}^t$ in the 3-dimensional simplex. Training with weights in (a) and testing with weights in (c) is supervised multi-source domain adaptation to domain 2 (d_2), while (b)-(c) is the unsupervised version, with no train data from d_2; training with weights in (a) and testing with weights in (d) is multi-domain learning, also illustrated with configurations (a)-(e) (train domain d_1 is not seen in test), and (b)-(d) (test domain d_2 is unseen in training).

This framework describes the real context closely because, in a typical setting in machine translation, we collect the most extensive collection of parallel data for the chosen language pair to achieve optimal performance for the task of interest. In such situations, the train data's distribution is opportunistic. The test data distribution is also a pre-determined mixture of the target domains. A key objective in training (multi-)domain

NMT models is to mitigate the detrimental effects of a possible mismatch between these distributions.

Train data may be a mixture of many domains such as the JRC-Acquis Communautaire corpus (LAW-domain) (Steinberger et al., 2006) or documentation for KDE, Ubuntu, GNOME and PHP from the Opus collection (Tiedemann, 2009) (IT-domain). We can also leverage a collection of data with no specific topic and genre, such as Paracrawl (Bañón et al., 2020), for example in Ng et al. (2019).

In (multi-)domain adaptation, the test data is a mixture of target domains. The weight of each target domain in the mixture is proportional to the priority of the domain. Empirically, test data consists of the test sets of the target domains. The performance of an MT system is the average of its performance on these tests weighted by domain weight in the mixture. Most of papers use uniform weights. However, we can assess the quality of a multi-domain MT system with different priorities over the target domains using weighted mean.

3.2.2 Four main sub-problems

From our study of the literature of (multi-)domain adaptation, we realize that all the cases of (multi-)domain adaptation can be classified into four groups by answering two following questions:

- Is/Are the train domain(s) determined?
- Is/Are the test domain(s) determined?

More precisely, the first question asks whether the train data is composed of a number of known domains. For example, the domain of a mixture of multiple corpora of specific topics, such as JRC-Acquis Communautaire corpus (LAW-domain) and KDE (IT-domain), is well defined. However, the domains in Paracrawl (Bañón et al., 2020) are unknown because the corpus was built by crawling the content of web-sites without specifying topic or purpose. The second question asks whether the domain of the test data is determined. If there exists a collection of text (monolingual or parallel) that defines the test domain, then it is domain-determined and vice versa.

The first case of (multi-)domain adaptation, which will be presented in Section 3.3, is supervised multi-domain adaptation in which domain labels are both available in training and testing. The format of train data is the triple (d, x, y), in which d is the index of the domain, (x, y) is a pair of parallel sentences. The format of test data is (d, x). This case might be the most straightforward situation. Many studies have been conducted in this setting (Pham et al., 2021).

The second case that we discuss in Section 3.4 considers the use of the parallel data

crudely collected from web sites such as Paracrawl (Bañón et al., 2020) or Commoncrawl [1]. The content of these corpora is a mixture of many topics, but unfortunately, there is not any available domain label for sentences. Fortunately, in this second case, the target domains are known, i.e., there exist data for these domains, which can be used to adapt the model. The format of train data will be $(?,x,y)$. The format of test data will be (d,x). This case focuses on the exploitation of opportunistic text.

The third case in Section 3.5 assumes the availability of train data correctly domain-labeled, whereas the test data can come from any possible domain. This case focuses on the robustness of the MT system against any potential shift distribution in testing. The format of train data is (d,x,y). The format of test data is $(?,x)$. The third setting is very close to most translations on the web, where the provenance of the input text is unknown.

Finally, the last setting in Section 3.6 focuses on both exploiting opportunistic data in training and being robust against unknown test distribution. The format of train data is $(?,x,y)$. The format of test data is $(?,x)$. We dedicate the four following sections to discuss each setting more thoroughly.

Recently the work of Chu & Wang (2018b) attempted to describe the landscape of MT domain adaptation by categorizing previous methods in 2 main classes: data-centric and model-centric. The data-centric category includes methods that manipulate the train distribution to resemble the distribution of the target domain better. The model-centric methods focus on changing the architecture, modifying the train objective, and improving the inference. Ramponi & Plank (2020) also adopted this taxonomy in their survey on unsupervised domain adaptation in natural language processing.

According to Chu & Wang (2018a), the data-centric methods focus on two paradigms, including 1) collecting parallel data related to the domain target, 2) creating synthetic data resembling the domain target. The first paradigm searches for similar examples to the ones of the domain target to enlarge the target domain's train data. The second paradigm aims to create pseudo samples resembling the data of the domain target. Besides the two paradigms, we propose another paradigm, which is data sampling. The data sampling paradigm consists of changing the data sampling scheme during the training course to mitigate the heterogeneity of the data size between domains and the variety of the "difficulty" of the domains. The data-centric group consists of 2 paradigms, including 1) data sellection, 2) data synthesis.

This taxonomy is primarily adopted in MT domain adaptation's research. However, we find a naivety in this classification as it misses delivering an answer for the most ultimate question: "which method solves which problem?". The four following sections

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

will explain how the model-centric and the data-centric categories solve 4 (multi-)domain adaptation cases. We also introduce several adaptation problems that need more research effort and other applications of existing methods. The work of Pham et al. (2021) recently gave a brief review of several well-known adaptation methods via a reevaluation with different domain adaptation cases. However, the experiments are still limited in the first case 3.3 and the third case 3.5 as they excluded non-domain-determined train data such as crawled corpora in their experiments.

3.3 Supervised (multi-)domain adaptation

In the supervised (multi-)domain adaptation problem, the domain label is available in both the train data and the test data. Furthermore, the domain(s) in the test data is(are) included in the train data. In this problem, we would like to train a single model that performs the best in our target domains, given train data from those domains and probably from other domains. This setting is the most popular adaption problem in the literature. This first case represents the easiest requirement for an MT model. A model needs to achieve the best performance on the domains of its train data. The difficulty of this situation is to both exploit the proximity between domains while mitigating the interference due to inter-domain heterogeneity. Effectively, similar topics, such as LEGAL and ADMINISTRATIVE, might improve the vocabulary coverage of each other as both domains share the same domain-specific words. However, distant topics, such as RELIGION and IT might confuse the NMT system when sharing the same parameters. In the following sections, we will discover how model-centric methods and data-centric methods solve this case.

3.3.1 Model-centric approaches

In the case of supervised multi-domain adaptation, model-centric methods focus on adding domain-specific parameters to reduce the interference between domains while keeping the total number of parameters small. The simplest approaches use domain tags. For example, Kobus et al. (2017) proposed appending a special token to each source sequence indicating its domain such as $< Domain = IT >$ and train the NMT model with this input. However, this method requires the domain tag of a sentence before translating it. Therefore, we have to predict the domain tag of the source sentence if it is from an unknown origin. Britz et al. (2017) originally proposed appending domain tag to the target sequence so that the decoder will predict the domain in which it will generate the translation. Instead of using domain tags, (Kobus et al., 2017) proposed using domain embeddings to incorporate the domain information into the context of the translation. Kobus et al. (2017) concatenated a domain embedding of small size (e.g., 4) to the embedding of each token in the input sequence. Each domain has its own embedding. Instead of using the domain

embedding to represent the domain in the representation, Pham et al. (2019) used a sparse word embedding called "lexicalized domain representation", in which a number of dimension are reserved for one domain and will be nullified if the model translates in other domains. We will discuss more this method in Chapter 5. Besides domain embeddings, it is possible to use domain-specified layers that can be plugged between 2 consecutive layers of the NMT model without changing the architecture. There are 2 types of plug-in layers: 1) residual adapters (Bapna & Firat, 2019b; Pham et al., 2020a) and 2) hidden unit contribution layers (Vilar, 2018). Residual adapters were first introduced by Rebuffi et al. (2017) in computer vision. Bapna & Firat (2019b) proposed this fine-tuning paradigm for domain adaptation and for multilingual NMT. They introduced a variant of residual adapters introduced in Rebuffi et al. (2017) composed of 2 linear projections and *ReLU* activation function. The adapters are plugged into the NMT model as follows

$$h^l_{enc/dec} = h^l_{enc/dec} + ADAP^l_{enc/dec}(h^l_{enc/dec}) \tag{3.1}$$

where $ADAP^l_{enc/dec}$ is the adapter corresponding to the l^{th} layer of the encoder/the decoder. Pham et al. (2020a) studied the use of the residual adapters for multi-domain adaptation and propose several techniques, including the regularization, a gating mechanism, to improve the robustness of the model. They will be presented in Chapter 6. The hidden unit contribution layers (LHUC) were proposed by Vilar (2018) to adapt an NMT model to a domain. The author applied an LHUC layer to the model as follows

$$h^l_{enc/dec} = h^l_{enc/dec} \odot a(\rho^l_{enc/dec}) \tag{3.2}$$

where $\rho^l_{enc/dec}$ is the adapter corresponding to the l^{th} layer of the encoder/the decoder and $\rho^l_{enc/dec} \in \mathbb{R}^d$. $a(.)$ is a scaled element-wise sigmoid function.

$$a(x) = \frac{2}{1+e^{-x}}$$

Residual adapters and LHUC layers adapt a pretrained model without changing its parameters.

While the previous methods aim to discriminate domains to reduce the interference, Britz et al. (2017) were motivated to learn hidden representations that are invariant between domains. More precisely, the authors use a binary classifier that takes the output of the encoder as input to predict the domain of the source sequence. They inverse the sign of the gradient with respect to the loss of the classifier to confuse it, i.e., making the hidden representation of the encoder invariant between 2 domains. This technique is related to A-distance, which is a measure of similarity between two probability distribu-

tions. Ben-David et al. (2006) showed that the A-distance between the source and target distributions is a crucial part of an upper generalization bound for domain adaptation. The authors hypothesized that it should be difficult to discriminate between the source and target domains to transfer between them well. Zeng et al. (2018)'s work was also inspired by this idea. However, instead of forcing the encoder's output to be invariant between domains, the authors aimed to extract domain-agnostic features and domain-specific features from the output of the encoder and feeding these features to the decoder. To this end, they used two different non-linear transformations, which map the output of the encoder to two feature vectors of the same size. Then, they applied a domain classifier on each feature vector. The domain-agnostic feature vector is trained to confuse the classifier, whereas the domain-specific feature vector is trained to facilitate its classifier.

Michel & Neubig (2018) adapted a pretrained model to a multi-user personalized model by fine-tuning the bias of the output layer to each particular user of the MT system. The number of additional parameters is $|S| \times |\Sigma_y|$, in which S is the set of the users. The authors reduced the size of the bias matrix $B \in \mathbb{R}^{|S| \times |\Sigma_y|}$, where each row is bias vector for one user, by factoring it into lower dimension representation, i.e.

$$\begin{array}{rcl} B & = & S \times \tilde{B}, \\ & where & \\ S & \in & \mathbb{R}^{|S| \times r}, \\ \tilde{B} & \in & \mathbb{R}^{r \times |\Sigma_y|}. \end{array} \tag{3.3}$$

Jiang et al. (2020)'s work was inspired by the mixture of expert paradigm. Their model is based on the Transformer architecture (Vaswani et al., 2017) as they integrate a domain-mixing mechanism to the multi-head attention layer. As we explained in Section 2.5.1, each head of a multi-head attention layer is computed as follows

$$head_i = Attention\big(QW_i^Q, VW_i^V, KW_i^K\big). \tag{3.4}$$

Suppose that there are K domains: at the i^{th} head, for the query, the key and the value components, there are K transformation matrices $W_{i,j}^Q | j \in [1,K]$, $W_{i,j}^K | j \in [1,K]$, $W_{i,j}^V | j \in [1,K]$ respectively. The domain-mixing mechanism is applied to the query, the key and the value components as follows

$$\begin{array}{rcl} Q_i^t & = & \sum_{j=1}^K Q^t W_{i,j}^Q * D_j(x_t), \\ K_i^t & = & \sum_{j=1}^K K^t W_{i,j}^K * D_j(x_t), \\ V_i^t & = & \sum_{j=1}^K V^t W_{i,j}^V * D_j(x_t), \\ head_i & = & Attention\big(Q_i, K_i, V_i\big), \end{array} \tag{3.5}$$

where the subscript t indicates the position of the vector in the sequence of hidden representations, $D(x_t) \in \mathbb{R}^K$ is the domain proportion of the corresponding token x_t. The proportion of domains of a token is computed by a domain classifier, that takes the input embedding of that token as input. By doing this, the model adapts the hidden representation of a token to the domain to which the token likely belongs. Consequently, the model is able to discriminate domains for domain-specific tokens while transferring knowledge between domains for domain-agnostic tokens. However, the size of the model is proportional to the number of domains, which is not practical in the real applications. The idea can be applied to residual adapters or LHUC layers.

Recently, Gong et al. (2021b,a) proposed learning domain-specific head selections in the multi-head attention mechanism to mitigate the interference between domains/languages. The authors used a variational NMT model with discrete latent variables $z_t^{(h)}$ from a Bernoulli distribution indicating whether a task t selects the attention head h. The training jointly learns the NMT model and the inference networks $q_{\theta_t^{(h)}}(z)$ of the pairs of (task t, head h) by optimizing the evidence lower bound (ELBO) (Kingma & Welling, 2014) with Gumbel-softmax reparametrization trick (Jang et al., 2017).

Besides the model-centric methods related to the architecture, multi-task training is another solution to the supervised (multi-)domain adaptation problem.

We illustrate several well-known model-centric methods for the supervised domain adaptation problem in figure 3.2.

3.3.2 Data-centric approaches

3.3.2.1 Data sampling

In supervised multi-domain adaptation, data sampling approaches aim to balance the contribution of each domain to the final model. In multi-domain train data collected opportunistically, the data size of each domain varies from few thousands examples to millions examples. In a trivial mixture of data, small domains are usually excessively undersampled causing a sub-optimal performance in average. However, if we equally sample data from every domain, the NMT model is easily over-fitted in the small domains. Therefore, finding an optimal sampling scheme across domains is essential. Wang et al. (2020c) proposed parameterizing the probability of sampling data from a domain and learned this probability via REINFORCE algorithm Williams (1992) using rewards computed from the cosine-similarity between the gradient over in-domain train data and the gradient over dev-sets' data. We compare this technique to our approach in Chapter 7.

Other data-centric methods, that do not take into account the domains of the train data will be discussed in Section 3.4.2.

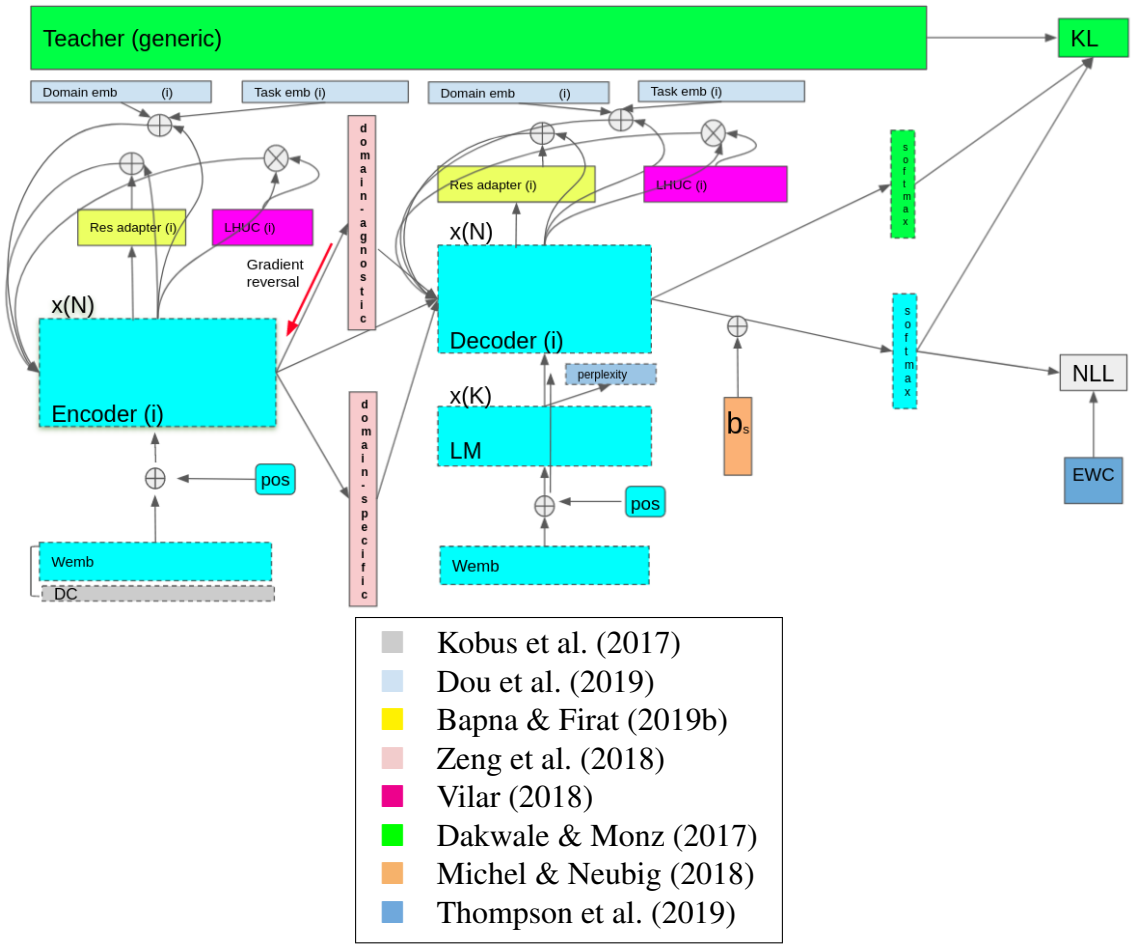


Figure 3.2: Each color except the blue corresponds to one model-centric method. The blue represents the NMT model.

3.4 Undetermined train domain, determined test domain

In this situation, the train data is composed by an unknown number of domains. The second case focuses on adapting the NMT model with an unknown source domain while the target domain is well defined. There are two situations: 1) there exists parallel data in the target domain 2) there exists only monolingual data in the target domain. The two following sections will discuss how each group of method solves these cases.

3.4.1 Model-centric approaches

First, we discuss the case with one target domain in which there exist parallel data. In this case, we can apply the same techniques proposed for supervised (multi-)domain adaptation by considering the source domain as a generic domain. Besides, fine-tuning is a very efficient approach for this problem(Luong & Manning, 2015; Servan et al., 2016; Freitag

& Al-Onaizan, 2016; Miceli Barone et al., 2017). We first train an NMT model with the mixture of source domains, then continue training this model with the parallel data of the target domain. According to a recent review of multi-domain adaptation conducted by Pham et al. (2021), fine-tuning is the strongest baseline in supervised domain adaptation. However, fine-tuned NMT models usually suffer from catastrophic forgetting (McCloskey & Cohen, 1989) as their performances drop dramatically in the source domains. To mitigate the catastrophic forgetting, several regularization techniques were introduced, including mixed fine-tuning (Chu et al., 2017), uniform weight-decay (Miceli Barone et al., 2017), elastic weight consolidation (EWC) (Kirkpatrick et al., 2016; Thompson et al., 2019; Saunders et al., 2019) and knowledge distillation (Dakwale & Monz, 2017).

Mixed fine-tuning (Chu et al., 2017) adapts an NMT model with the mixture of the source domain and the target domain (by oversampling the target domain). The method adapts the NMT model to the domain of interest thanks to the oversampling while maintaining its robustness to generic text as it continues to be trained with the source domain.

Weight decay (Miceli Barone et al., 2017) continues training the NMT model on the target domain's data with a regularized loss

$$L_{CE}(\theta, \mathscr{D}_e^t, g^t) + \alpha * \parallel \theta - \theta^A \parallel_{L_2}, \tag{3.6}$$

where θ^A is the value of the pretrained model, $\mathscr{D}_e^t, g^t$ characterize the target domain t. Fine-tuning with the new loss fits the model to the target domain while preserving the the old pretrained parameter values, therefore preventing the overfitting of the NMT model in the target domain and maintaining the generality over source domains. Kirkpatrick et al. (2016); Thompson et al. (2019); Saunders et al. (2019) were also motivated to penalize the changes of the parameters compared to the initial model. However, the authors argued that not every parameter has the same contribution to maintain the generality to the old domains and that we can tune the parameters unimportant for the old domains to the target domain. The contribution of each parameter of the pretrained model is approximated by the diagonal of the Fisher matrix computed over the data of the source domains. Fine-tuning with EWC uses the following loss

$$L_{CE}(\theta, \mathscr{D}_e^t, g^t) + \sum_i \frac{\lambda}{2} F_i * (\theta_i - \theta_i^A)^2, \tag{3.7}$$

where the F_i is the i^th element of the diagonal of the Fischer matrix approximated as follows

$$\bar{F} = E_{x \sim \mathscr{D}_e^s, y \sim g^s(x)} \left[\nabla logP(y|x,\theta)_{|\theta^A} \nabla logP(y|x,\theta)_{|\theta^A}^T \right], \tag{3.8}$$

in which $\mathscr{D}_e^s, g^s$ characterize previous train domains. Dakwale & Monz (2017)'s method

was motivated by the knowledge distillation paradigm (Hinton et al., 2015). The authors proposed regularizing the standard cross-entropy loss with the Kullback-Leibner distance (Kullback & Leibler, 1951) between two predicting distributions produced by the old model and the new model as follows

$$L_{CE}(\theta, \mathscr{D}_e^t, g^t) + \alpha * E_{x\sim\mathscr{D}_e^s, y\sim g^s(x)}\big[KL(P(.|x,\theta)|P(.|x,\theta_A))\big], \tag{3.9}$$

in which $\mathscr{D}_e^s, g^s$ characterize previous train domains. Instead of continuing training with target domain only, Chen et al. (2017) differentiated directly domain-relevant instances and irrelevant instances via instance weighting. The authors compute the weight of each instance by a domain classifier, that is trained with source sequences. The training will maximize the following objective

$$\hat{\theta} = \arg\max_{\theta} E_{x\sim\mathscr{D}_e^s, y\sim g^s(x)}(1+p_d(x))log(P(y|x,\theta)) + E_{x\sim\mathscr{D}_e^t, y\sim g^t(x)}(1+p_d(x))log(P(y|x,\theta)) \tag{3.10}$$

where $\mathscr{D}_e^t, g^t$, $\mathscr{D}_e^s, g^s$ are the target domain and other domains, $p_d(x)$ is the probability that x comes from the target domain. Wang et al. (2017b) proposed using different domain-relevance metrics for instance weighting.

Besides methods using auxiliary losses, ensemble methods are also promising. For example, Freitag & Al-Onaizan (2016) proposed ensembling the pretrained model and the fine-tuned model to combine the advantage of both models: the specialization in the target domain and the generalization over general text.

Still in the case of uni-domain adaptation, but without parallel data, the model-centric approaches mostly use monolingual data in the target language of the target domain. The proposed methods mostly adapted the decoder to the target domain. For example, Gülçehre et al. (2015) proposed training a language model adapted to the target domain and fusing the language model to the decoder. The fusion could be deep or shallow. The deep fusion combined the hidden representation of the decoder and the one of the language model before computing the prediction probability. The shallow fusion combined the prediction probability computed by the decoder and the one computed by the language model. Domhan & Hieber (2017) was also motivated by this idea. However, the authors proposed jointly training the language model and the NMT model via multi-task training. Furthermore, the decoder and the language model shared the word embedding of the target side.

All previous methods were motivated to adapt an NMT model to a specific domain. We realize that they can hardly be applied to multi-domain adaptation because all the parameters of the MT model are adapted to one domain. However, in the case where there

are parallel data of the target domains, we could use model-centric methods proposed in the supervised adaptation by considering the train domain a generic domain. Recently, Dou et al. (2019) proposed using domain embeddings and task embeddings to adapt an NMT model to the target domain using reconstruction loss on the monolingual data. More precisely, for each layer l^{th} of an ANMT model, there are 2 task embeddings $\theta^{\gamma,l}_{task}$, $\gamma \in \{MT, LM\}$, which correspond to translation task and language modeling task respectively. Furthermore, for each layer l^{th}, and for each domain d, there is a domain task $\theta^{d,l}_{dom}$. The layer l^{th} of encoder/decoder will be as follows

$$h^l = LAYER^l(h^{l_1}) + \theta^{d,l}_{domain} + \theta^{\gamma,l}_{task} \tag{3.11}$$

Now, the parallel data will be used to compute the translation loss, while the monolingual data is used to compute the language modeling loss. The authors proposed adding noises to the source sequence while computing the LM loss. Using domain-specific embeddings enables the model to be adapted to multiple domains at once. Despite being applied to multi-lingual machine translation, the monolingual adapter proposed by Philip et al. (2020) shares the same spirit and can be applied to this situation.

3.4.2 Data-centric approaches

According to our study, the previous data-centric approaches proposed to this situation belong to all three paradigms, including data selection, data synthesis, and data sampling. The two following sections will discuss the methods of each paradigm.

3.4.2.1 Data selection

Data selection approaches collect parallel data, which resemble the target domain. The selection is usually based on a score of proximity between a parallel example and the domain. The score of proximity can be computed via sentence embeddings or variants of Moore & Lewis (2010) score. For example, given two corpora of a target domain $D_{I\text{-}src}$, $D_{I\text{-}tgt}$ and two corpora of the source domain $D_{O\text{-}src}$, $D_{O\text{-}tgt}$, Axelrod et al. (2011) computed a bilingual version of Moore & Lewis score of a sentence pair as follows

$$S_{bi}(x,y) \quad = \quad H_{I\text{-}src}(x) - H_{O\text{-}src}(x) + H_{I\text{-}tgt}(y) - H_{O\text{-}tgt}(y), \tag{3.12}$$

where the cross-entropy $H_*(z)_{|*\in[I\text{-}src,I\text{-}tgt,O\text{-}src,O\text{-}tgt],z\in[x,y]}$ of the sentence z is computed by a language model trained only with the corpus D_*. Duh et al. (2013) proposed the same formulation as proposed Axelrod et al. (2011) but used a neural language model instead of a statistic language model. In a survey of data selection methods for neural machine

translation, Silva et al. (2018) evaluated 3 popular methods in the domain adaptation task, including the cross-entropy difference, the Term Frequency-Inverse Document Frequency embeddings (TF-IDF) (Salton & Yang, 1973) and the Feature Decay algorithm (FDA) (Poncelas et al., 2018). More precisely, the cross-entropy difference was computed as described above and normalized by sentence length. To compute a TF-IDF representation vector, Silva et al. (2018) consider each sentence of the target domain as a query and every sentence in the source domain as a key. The tf-idf vectors of the queries and the keys are computed as in Salton & Yang (1973). The score of proximity between a query and a key is the cosine-similarity of theirs tf-idf vectors. Based on the score of proximity, for each sentence of the target domain, we retrieve K nearest-neighbors in the source domain. The collection of the retrieved sentences is the result of the method. Finally, FDA aims to extract from train data a set of sentences that are most relevant to the test set of the target domain. We call an extracted n-gram a feature. The method extracts n-grams from the source side of the test set. In the beginning, the features are assigned the same value. Each sentence of the train data is scored as the normalized (by dividing by the number of words) sum of the values of its features. Then, the method selects the sentence with the highest score and adds it to the set of selected data (which initially is empty). After selecting a sentence, the values of the features contained in it are decreased. The decay function is defined as follows

$$decay(f) = init(f) * 0.5^{C_L(f)}, \tag{3.13}$$

where f is an n-gram, $init(f)$ is the initial value assigned to each feature and $C_L(f)$ is the count of the feature f in the selected data.

Wang et al. (2017a) proposed using a sentence embedding to represent a sentence instead of a tf-idf vector. For each language side (source/target) the authors computed the centroid of the target domain and one of the source domain. Assume $C_{E_{in}}$, $C_{E_{out}}$ are the centroid of the target domain and the source domain in the source language, $C_{F_{in}}$, $C_{F_{out}}$ are the centroid of the target domain and the source domain in the target language, v_e is the sentence embedding of a source sentence e, v_f is the sentence embedding of a source sentence f, then the proximity of the example (e,f) to the target domains is defined as follows

$$d(v_e, C_{E_{in}}) - d(v_e, C_{E_{out}}) + d(v_f, C_{F_{in}}) - d(v_f, C_{F_{out}}), \tag{3.14}$$

where $d(.,.)$ is the Euclidean distance in $\mathbb{R}^d$. Aharoni & Goldberg (2020) proposed using sentence embeddings computed by a pretrained Bidirectional Encoding Representation Transformer (BERT) and the cosine-similarity to retrieve domain-related examples.

3.4.2.2 Data synthesis

The most efficient approach in this paradigm is backtranslation (Sennrich et al., 2016a), which consists of translating the monolingual data of the target language to the source language. Burlot & Yvon (2018) showed significant improvement of an NMT model in the target domain when trained with a mixture of parallel data and in-domain back-translated data. Without backtranslating the target-side data, Currey et al. (2017) created artificial sentence pairs from the monolingual data in the target language so that each source sentence is identical to the target sentence. Training an NMT model with the mixture of parallel data and artificial sentences improves the accuracy of the translation on named entities and other words that should remain identical between the source and target languages. Recently Currey et al. (2020) proposed distilling knowledge from multiple domain experts to a multi-domain NMT model. The authors translated each source side in-domain corpus with the corresponding adapted model, then mixed the artificial corpora to the train corpora and continued training the multi-domain NMT model with the new corpora.

3.4.2.3 Data sampling

Data sampling methods dynamically change the composition of the train data over time. In practice, the evolution of the train data is beneficial for training an NMT model specialized to a domain. For example, in the fine-tuning approaches, the training begins with every available data and finishes with the data of the target domain. Data sampling methods consist of building an automatic curriculum without human supervision. For example, van der Wees et al. (2017) proposed gradual fine-tuning, which first computes a sampling distribution based on the cross-entropy difference (CED) (3.12) of each example, then gradually decreases the number of sampled examples after each epoch. Formally, the CED score of each example is normalized as follows

$$\tilde{CED}(x) = 1 - \frac{CED(x) - min(CED_G)}{max(CED_G) - min(CED_G)}, \tag{3.15}$$

where G is the train corpus. Therefore higher-ranked examples have a higher $\tilde{CED}$ score rather than having lower CED as in Axelrod et al. (2011). The sampling distribution is computed as follows

$$\omega(x) = \frac{\tilde{CED}(x)}{\sum_{x' \in G} \tilde{CED}(x')}. \tag{3.16}$$

For each epoch i^{th}, a number n_i of examples are selected according to the previous distribution. n_i is defined as follows

$$n_i = \alpha \cdot |G| \cdot \beta^{\lfloor \frac{i-1}{n}, \rfloor} \tag{3.17}$$

where $\alpha \in [0,1]$ is the relative start size, $\beta \in [0,1]$ is the retention rate. Via the same mechanism, Wang et al. (2019) proposed a more sophisticated dynamical sampling scheme combining two scores of an example, $domain - CED$ and $noise - CED$. The authors proposed two variants: mixed co-curriculum, which scores an example by the sum of its CED scores, and cascaded co-curriculum, which first selects examples by their $domain - CED$ scores then retains top examples according to their $noise - CED$ scores from the previous selection. Furthermore, Wang et al. (2019) proposed to recompute the language model of the noisy data for each epoch. Instead of increasing the domain-relevance of the train data, Zhang et al. (2019) did the opposite. More precisely, they reordered the train data according to their relevance to the target domain then equally split the whole corpus into many shards containing samples of a similar score. They trained an NMT model with one shard per epoch in a decreasing order of domain-relevance.

Besides well-known metrics for domain-relevance, Zhang et al. (2019) proposed a parameterized scorer, that evaluates the usefulness of each sample to the performance on the target domain, and optimized its parameters via Bayesian optimization. More precisely, the scorer was formulated as follows

$$f(x,y) = V \cdot \mathbf{F}(x,y), \tag{3.18}$$

where the feature vector $\mathbf{F}(x,y)$ was extracted from the example, and the weight vector V was learned by Bayesian optimization. Each element in $\mathbf{F}(x,y)$ represents the relevance of the example to a target domain. Once the scorer was optimized, the training was conducted as in van der Wees et al. (2017); Wang et al. (2019).

3.5 Determined train domain, undetermined test domain

The third case is interesting because it resembles the real context of machine translation's applications. Effectively, the users' text can be from any possible topic or for any possible purpose (genre). In this situation, the NMT model needs to be both robust to the unseen domains and adapted to known domains.

3.5.1 Model-centric approaches

Mixture models are a promising solution for this situation as they combine domain-adapted systems to perform the translation. Effectively, the performance of mixture models is guaranteed in the source domains while using a convex combination of the adapted systems is robust against unseen domains. Mixture models have been successfully applied for SMT models (Sennrich, 2012a,b; Carpuat et al., 2014). Freitag & Al-Onaizan (2016); Sajjad et al. (2017); Saunders et al. (2019) applied mixture models to NMT models. The contribution of each adapted model to the combination was uniform in Freitag & Al-Onaizan (2016) while Sajjad et al. (2017) pre-finetuned the weights by Bayesian optimization on a development set. However, a heuristic static weights are sub-optimal when the test domain is highly variable. Saunders et al. (2019) proposed computing the weights of the mixture for each source sentence x at the i^{th} decoding step as follows

$$\begin{aligned} W_{k,i} &= \sum_t P(t=k|h_i,x)\lambda_{k,t}, \\ &\textit{where} \\ P(t=k|h_i,x) = P(t=k|x) &= \frac{P^k_{LM}(x)}{\sum_{k'} P^{k'}_{LM}(x)}, \end{aligned} \tag{3.19}$$

where $P^{k'}_{LM}(x)$ is the probability of the source sentence x according to a language model learned from domain k', $\lambda_{k,t}$ can be uniform, identity or pre-finetuned with a development set. Saunders et al. (2019) also proposed varying the mixture's weights during the inference by conditioning the domain posterior probability on both x and h_i as follows

$$P(t=k|h_i,x) = \frac{P(h_i|t,x)P(t|x)}{\sum_{k'} P(h_i|k',x)P(k'|x)}. \tag{3.20}$$

Besides one might be more interested in domain robustness than in domain specialization. The mixture model does not include the domain robustness in the training objective. Müller et al. (2020) discussed several regularization methods to mitigate the problem, including the subword regularization (Kudo, 2018), the defensive distillation (Papernot et al., 2016), the reconstruction (Tu et al., 2017) and the neural noisy channel reranking (Li & Jurafsky, 2016). The distributional robustness optimization (Ben-Tal et al., 2013; Oren et al., 2019) is also a promising paradigm as the related methods optimize models so that they perform well over a wide range of potential test distributions. However, the application of this paradigm to neural machine translation has not yet been explored.

3.5.2 Data-centric approaches

The data-centric paradigm can not be applied to this situation since the domain of the test sentences is unknown. Indeed, the data-centric approaches aim to build train data that approximates a target domain and require the monolingual data of that domain to create the pseudo in-domain data. That can only be done when we know the target domain.

3.6 Undetermined train domain, undetermined test domain

3.6.1 Model-centric approaches

Farajian et al. (2017b); Li et al. (2018) proposed fine-tuning on-the-fly a pretrained model with a mini-batch similar to the source sentence before translating it. The authors chose the learning rate and the number of fine-tuning iterations according to the similarity score between the retrieved mini-batch and the source sentence so that the higher similarity the higher the learning rate, the more iterations. Bulté & Tezcan (2019); Bapna & Firat (2019a); Pham et al. (2020b); Xu et al. (2020) proposed learning an NMT model to reuse parallel examples whose source sentence is similar to the source sentence. Some of these techniques are discussed in Chapter 8.

3.6.2 Data-centric approaches

The data-centric paradigm can not be applied to this situation for the same reason as in previous Section 3.5.2

3.7 Multi-domain and multi-lingual machine translation

We can put multi-domain and multilingual machine translation under the same umbrella of multi-task learning where a task is one language pair or one domain. Most of the model-centric approaches can be applied to both of these tasks. For example, the target-language prefix token used in one-to-many multilingual NMT model (Johnson et al., 2017a; Aharoni et al., 2019) has a similar role as the domain tag used in multi-domain NMT (Kobus et al., 2017). Bapna & Firat (2019b) used the residual adapters for both domain-adaptation and multilingual NMT. Multi-domain NMT and multilingual NMT aim to improve low-resourced domain/language quality by exploiting the proximity between the low-resourced target domain/language and the high-resourced domains/languages. Gong et al. (2021b) learned domain/language-specific head selection to improve the multi-domain/multilingual NMT system. In the data-centric category, Wang et al.

(2020c)'s method can be used for both problems.

However, at a closer look, multilingual MT has a different context than multi-domain MT. Multilingual MT does not have the same languages on both sides and aims to perform zero-shot translations, in which the pair of source and target languages are not included in the training data, thanks to a positive knowledge transfer via pivot languages. In contrast, multi-domain MT focuses on one language pair and exploits the positive knowledge transfer between the domains.

In summary, both problems rest on the hypothesis on the positive transfer of multi-task learning. Many approaches can be applied to both problems. However, the natures of these tasks are different. Thus the implementation of an approach might need an adaptation to each problem.

3.8 Conclusions

We want to summarize this chapter in Figure 3.3.

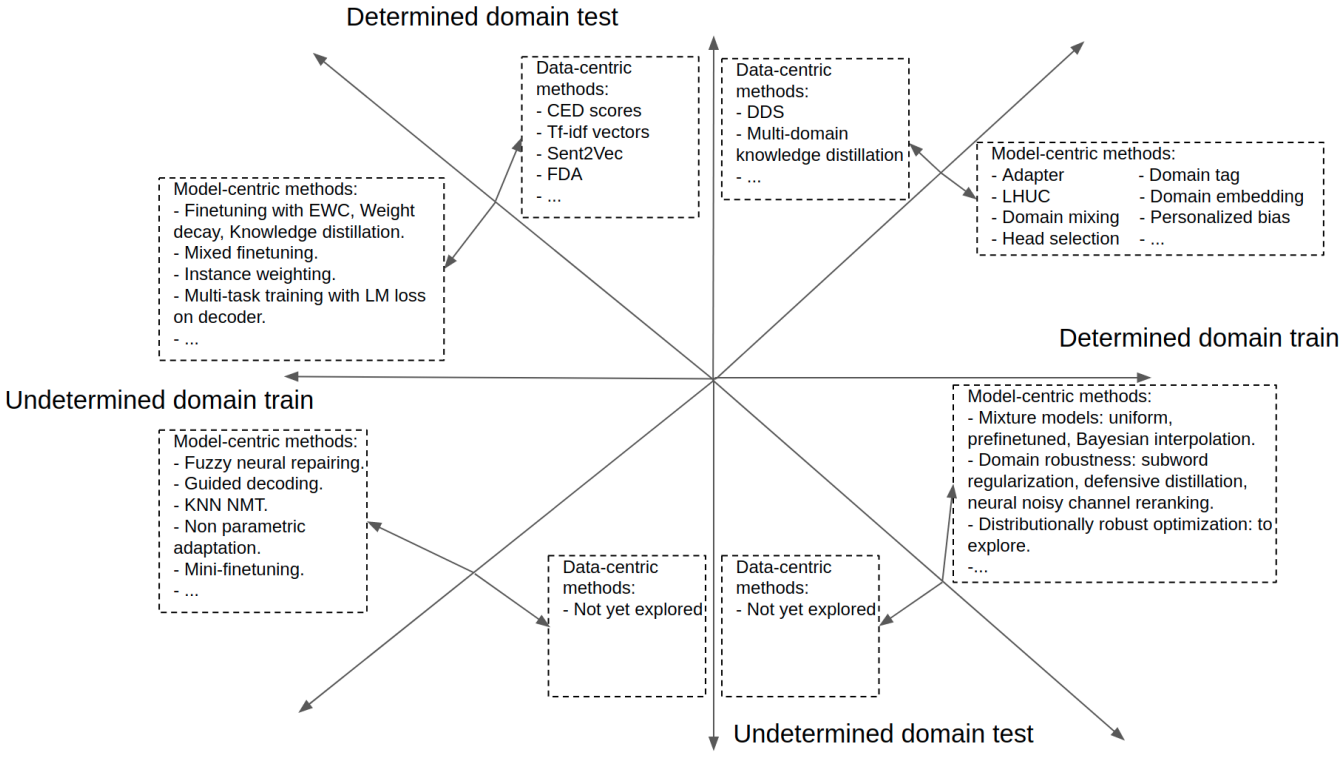


Figure 3.3: A complete overview of multi-domain adaptation with the four primary settings and the two groups of approaches.

This thesis mainly focuses on the first setting in which domain train and domain test are determined. We will present our most important contribution to the study in this setting in the next chapter. This work presents a novel multi-criteria evaluation, which requires

five primary properties for a multi-domain system, and reevaluates a wide range of popular model-centric methods with our experimental setting designed to evaluate these five criteria. In Chapter 8, we study different approaches for the last setting in which domain train and domain test are unspecified.

Chapter 4

Revisiting supervised multi-domain machine translation

4.1 Introduction

In this chapter, we analyze a wide range of popular model-centric methods which aim to solve supervised (multi-)domain adaptation (see definition in Section 3.3). First we formulate the motivations for developing multi-domain machine translation (MDMT) systems and the associated expectations with respect to performance. We define five essential requirements that an effective MDMT system should meet (Section 4.2). We reevaluate a wide range of popular MDMT systems and show that most of these expectations are hardly met. We suggest that further work is needed to analyze the current behavior of multi-domain systems better and to make them fully hold their promises.

Data-based Machine Translation (MT), whether statistical or neural, rests on well-understood machine learning principles. Given a train sample of matched source-target sentence pairs (x, y) drawn from an underlying distribution $\mathscr{D}_e$ of the input space Ω_e and a labeling function $g : \Omega_e \rightarrow \Omega_f$ where Ω_e and Ω_f are the set of sentences of the language e and language f respectively. An NMT model parameterized by θ (here, a translation function h_θ) is trained by minimizing the empirical expectation of a loss function $\ell(h_\theta(x), y)$. This approach ensures that the translation loss remains low when translating more sentences drawn from the same distribution. Owing to the great variability of language data, this ideal situation is rarely met in practice, warranting the study of an alternative scenario, where the test distribution $\mathscr{D}_e^T$ differs from train distribution $\mathscr{D}_e^S$ and the test labeling function g^T differs from the train labeling function g^S. In this setting, *domain adaptation* (DA) methods are in order.

Multi-domain (MD) machine translation (Sajjad et al., 2017; Farajian et al., 2017b; Kobus et al., 2017; Zeng et al., 2018; Pham et al., 2019) generalizes the conventional setting of domain mismatch by considering a mixture of multiple domains (i.e., multiple underlying distributions and multiple labeling functions). MDMT focuses on training

one single system using a mixture of multiple train domains and evaluating it with data from multiple test domains. MDMT corresponds to a very common situation, where all available data, no matter its origin, is used to train a robust system that performs well for any kind of new input. If the intuitions behind MDMT are quite simple, the exact specifications of MDMT systems are rarely spelled out: for instance, how should MDMT handle the heterogeneity of the domains' size? Should MDMT be robust to the intra-domain heterogeneity? Should MDMT also be robust to new domains? How should it exploit the proximity between domains? How should it handle domain labeling errors during the inference? How should it handle the growing number of domains?

Multi-domain seems more challenging than domain adaptation as it tries to optimize MT performance for a more diverse set of potential inputs, with an additional uncertainty regarding the distribution of test data. However, are there still situations where MDMT systems can surpass single domain adaptation, as is sometimes expected?

The first contribution is thus of methodological nature and consists of lists of expected properties of MDMT systems and associated measurements to evaluate them (Section 4.3). In doing so, we also shed light on new problems that arise in this context, regarding, for instance, the integration of new domains in the course of training or the computation of automatic domain tags. The second main contribution is experimental and consists in a thorough reanalysis of eight recent multi-domain model-centric approaches from the literature. We show in Section 4.5 that existing approaches still fail to match many of these requirements, notably with respect to the handling of a large number of heterogeneous domains and to dynamically integrating new domains in training.

This chapter draws from the following publication: Pham et al. (2021).

4.2 Requirements of multi-domain MT

In this section, we recap the main reasons for considering a multi-domain scenario and discuss their implications in terms of performance evaluation.

4.2.1 Formalizing multi-domain translation

As introduced in Section 3.2.1, a domain d of a language pair (e, f) is defined by an underlying distribution $\mathscr{D}_e^d$ over the space Ω_e of sentences in language e and a translation function $g^d : \Omega_e \rightarrow \Omega_f$, in which Ω_f is the space of sentences in language f. A typical learning scenario in MT is to have access to samples from n_d domains, which means that the train distribution on the source language $\mathscr{D}_e^S$ is a mixture $\mathscr{D}_e^S(x) = \sum_d \lambda^s(d)\mathscr{D}_e^d(x)$, with $\{\lambda^s(d), d = 1 \dots n_d\}$ the corresponding mixture weights ($\sum_d \lambda^s(d) = 1$). Multi-domain learning, as defined in Dredze & Crammer (2008) further assumes that domain

tags are also available in testing; the implication being that the test distribution is also as a mixture $\mathscr{D}_e^T(x) = \sum_d \lambda^t(d)\mathscr{D}_e^d(x)$ of several domains, making the problem distinct from mere domain adaption. A multi-domain learner is then expected to use these tags effectively (Joshi et al., 2012) when computing the combined translation function $g^d(x)$, and to perform well in all domains (Finkel & Manning, 2009). This setting is closely related to the multi-source adaptation problem formalized in Mansour et al. (2009a,b); Hoffman et al. (2018).

This definition seems to be the most accepted view of a multi-domain machine translation[1] and one that we also adopt here. Note that in the absence of further specification, the naive answer to the MDMT setting should be to estimate one translation function $\hat{g}^d(x)$ separately for each domain, then to translate using $\hat{g}(x,d) = \sum_{d'} g^{d'}(x)\mathbb{I}(d' = d)$, where $\mathbb{I}(x)$ is the indicator function. We now discuss the arguments that are put forward to proceed differently.

4.2.2 Reasons for building MDMT systems

The first motivation for moving away from the "one domain / one model" solution is to be practical in an environment diversified in terms of topics, genres, and styles (Sennrich et al., 2013; Farajian et al., 2017a). When faced with inputs that are potentially from multiple domains, it is easier and computationally cheaper to develop one system instead of optimizing and maintaining numerous engines. Multilingual machine translation also shares the same spirit (Johnson et al., 2017b). The underlying assumption here is that the number of domains of interest can be large, for example, fully personalized machine translation systems (Michel & Neubig, 2018).

The second reason rests on the linguistic properties of the translation function. The domain specificities are mainly expressed lexically and will primarily affect content words or multi-word expressions. On the other hand, the function words are domain agnostic and tend to remain semantically stable across domains, motivating some cross-domain parameter sharing. An MDMT system should simultaneously learn lexical domain peculiarities and leverage cross-domain similarities to improve the translation of generic contexts and words (Zeng et al., 2018; Pham et al., 2019; Jiang et al., 2020). We expect the MDMT scenario to be more profitable when the domain mixture includes similar domains that share more information.

The third motivation is of statistical nature. The train data available for each domain is usually unevenly distributed; domains such as BANK in English-German contain only a few thousand examples 6.3.1. Moreover, for some test domains, there may even be

[1] An exception is (Farajian et al., 2017b), where test translations rely on similarity scores between test and train sentences, rather than on domain labels.

no train data at all (Farajian et al., 2017a). Tuning an NMT model to a small dataset without regularizations usually gives us statistically less reliable estimates of the domain's parameters resulting in a high variance in the prediction. Training mix-domain systems likely reduces this variance at the expense of the statistical bias (Clark et al., 2012). Under this view, MDMT would be especially beneficial for domains with little train data. In the case of multilingual MT from English, a significant improvement for under-resourced languages was reported due to positive transfer, at the cost of a decrease in performance for well-resourced languages (Arivazhagan et al., 2019).

Ensembling multiple domain-specific MT models can also be justified for the sake of distributional robustness (Mansour et al., 2009a,b; Sennrich, 2012b,a; Carpuat et al., 2014; Freitag & Al-Onaizan, 2016; Sajjad et al., 2017; Saunders et al., 2019), for instance, when the test mixture differs from the train mixture or when it includes new domains unseen in training. An even more challenging case is when an MT system needs to perform well for any test distribution, as studied for SMT models in Huck et al. (2015) or language models in Oren et al. (2019). In all these cases, mixing domains in training will likely improve robustness against unexpected or adversarial test distribution.

Another motivation is that mixing domains can have a positive regularization effect for all domains. Introducing variability in training prevents DA from overfitting the available adaptation data and could help improve generalization even for well-resourced domains. Joshi et al. (2012) explored a related case, which shows that part of the benefits of MD training is due to an ensembling effect, where systems from multiple domains are simultaneously used in the prediction phase; this effect may persist even in the absence of clear domain separations.

In summary, there are multiple reasons for adopting MDMT, some already used in DA settings, and some original. These arguments are not mutually exclusive; however, each yields specific expectations for the performance of the MDMT models and should also require an appropriate evaluation procedure. If the motivation is primarily computational, then a drop in MT quality for multiple individual domains might be acceptable for the computational savings. If the objective is to improve statistical estimation, we expect that MDMT improves, at least for some under-resourced domains, over individually trained systems. Finally, if the goal is to make the system more robust to adversarial test distributions, one should use this setting to evaluate MDMT. The following section discusses ways that could challenge these requirements of MDMT systems.

4.3 Challenging multi-domain systems

In this section, we propose seven operational requirements that can be expected from an effective multi-domain system, and discuss ways to evaluate whether these requirements are actually met. All these evaluations will rest on comparison of translation performance, and do not depend on the choice of a particular metric. To make our results comparable with the literature, we will only use the BLEU score (Papineni et al., 2002) in Section 4.4, noting it may not be the best yardstick to assess subtle improvements of lexical choices that are often associated with domain adapted systems (Irvine et al., 2013). Other important figures of merit for MDMT systems are the computational training cost and the total number of parameters.

4.3.1 Multi-domain systems should be effective

A first expectation is that MDMT systems should perform well in the face of mixed-domain test data. We thus derive the following requirements.

[P1-LAB] A MDMT should perform better than the baseline which disregards domain labels. Evaluating this requirement is a matter of a mere comparison, assuming the test distribution of domains is known: if all domains are equally important, performance averages can be reported; if they are not, weighted averages should be used instead.

[P2-TUN] Additionally, one can expect that MDMT will improve over fine-tuning (Luong & Manning, 2015; Freitag & Al-Onaizan, 2016), at least in domains where data is scarce, or in situations where several domains are close. To evaluate this, we perform two measurements, using a real as well as an artificial scenario. In the real scenario, we simply compare the performance of MDMT and fine-tuning for domains of varying sizes and expect an important improvement for smaller domains in MDMT compared to fine-tuning. In the artificial scenario, we split a single domain in two parts which are considered distinct in training. The expectation here is that a MDMT should yield a clear gain for both pseudo sub-domains, which should benefit from the supplementary amount of relevant train data. In this situation, MDMT should even outperform fine-tuning on either of the pseudo sub-domain. The property we assess here is the robustness to the cross-domain heterogeneity.

4.3.2 Robustness to fuzzy domain separations

A second set of requirements is related to the definition of a domain. As repeatedly pointed out in the literature, parallel corpora in MT are often collected opportunistically

and the view that each corpus constitutes a single domain is often a gross approximation.[2] MDMT should aim to make the best of the available data and be robust to domain assignments. To challenge these requirements we evaluate the following requirements.

[P3-HET] The notion of a domain being a fragile one, an effective MDMT system should be able to discover not only when cross-domain sharing is useful (cf. requirement [P2-TUN]), but also when intra-domain heterogeneity is hurting. This requirement is tested by artificially conjoining two closely related domains into one during training, hoping that the loss in performance with respect to the original setting (using correct domain tags) will remain small. The property we assess here is the robustness to the intra-domain heterogeneity.

[P4-ERR] MDMTs should perform best when the true domain tag is known, but deteriorate gracefully in the face of tag errors; in this situation, catastrophic drops in performance are often observed. This requirement can be assessed by translating test texts with erroneous domain tags and reporting the subsequent loss in performance. The property we assess here is the robustness to the unseen domains.

[P5-UNK] A related situation occurs when the domain of a test document is unknown. Several situations need be considered: for domains seen in training, using automatically predicted domain labels should not be much worse than using the correct one. For test documents from unknown domains (zero-shot transfer), a good MD system should ideally outperform the default baseline that merges all available data. The property we assess here is the robustness to the erroneous domain tags.

[P6-DYN] Another requirement, more of an operational nature, is that an MDMT system should smoothly evolve to handle a growing number of domains, without having to retrain the full system each time new data is available. This is a requirement [P6-DYN] that we challenge by dynamically changing the number of train and test domains. The property we assess here is the robustness to the growing number of domains.

4.3.3 Scaling to a large number of domains

[P7-NUM] As mentioned above, MDMT systems have often been motivated by computational arguments. This argument is all the more sensible as the number of domains

[2] Two of our own "domains" actually comprise several subcorpora (IT and MED), see details in Section 4.4.1.

increases, making the optimization of many individual systems both ineffective and undesirable. For lack of having access to corpora containing very large sets (eg. in the order of 100-1000) domains, we experiment with automatically learned domains.

4.4 Experimental settings

4.4.1 Data and metrics

We experiment with translation from English into French and use texts initially originating from 6 domains, corresponding to the following data sources: the UFAL Medical corpus V1.0 (MED)[3], the European Central Bank corpus (BANK) (Tiedemann, 2012); The JRC-Acquis Communautaire corpus (LAW) (Steinberger et al., 2006), documentations for KDE, Ubuntu, GNOME and PHP from Opus collection (Tiedemann, 2009), collectively merged in a IT-domain, Ted Talks (TALK) (Cettolo et al., 2012), and the Koran (REL). Complementary experiments also use v12 of the News Commentary corpus (NEWS). Most corpora are available from the Opus web site.[4] These corpora were deduplicated and tokenized with in-house tools; statistics are in Table 4.1. To reduce the number of types and build open-vocabulary systems, we use Byte-Pair Encoding (Sennrich et al., 2016b) with 30,000 merge operations on a corpus containing all sentences in both languages.

We randomly select in each corpus a development and a test set of 1,000 lines and keep the rest for training.[5] Validation sets are used to chose the best model according to the average BLEU score (Papineni et al., 2002).[6] Statistical significance is estimated using bootstrap resampling (Koehn, 2004), implemented in compare-mt[7] (Neubig et al., 2019). We report significant differences at the level of $p = 0.05$.

We measure the distance between domains using the $\mathscr{H}$-Divergence (Ben-David et al., 2010b), which relates domain similarity to the test error of a domain discriminator: the larger the error, the closer the domains. Our discriminator is a SVM independently trained for each pair of domains, with sentence representations derived via mean pooling from the source side representation of the generic Transformer model. We used the scikit-learn[8] implementation with default values. Results in Table 4.2 show that all domains are well separated from all others, with REL being the furthest apart, while TALK is slightly more central.

[3] `https://ufal.mff.cuni.cz/ufal_medical_corpus`. We only use the in-domain (medical) subcorpora: PATR, EMEA, CESTA, ECDC.

[4] `http://opus.nlpl.eu`

[5] The code for reproducing our train, dev and test datasets is available at `https://github.com/qmpham/experiments`.

[6] We use detokenized, truecasing and the `multibleu` script.

[7] `https://github.com/neulab/compare-mt`

[8] `https://scikit-learn.org`

	MED	LAW	BANK	IT	TALK	REL	NEWS
# lines	2609 (0.68)	501 (0.13)	190 (0.05)	270 (0.07)	160 (0.04)	130 (0.03)	260 (0)
# tokens	133 / 154	17.1 / 19.6	6.3 / 7.3	3.6 / 4.6	3.6 / 4.0	3.2 / 3.4	7.8 / 9.2
# types	771 / 720	52.7 / 63.1	92.3 / 94.7	75.8 / 91.4	61.5 / 73.3	22.4 / 10.5	77.8 / 80.4
# uniq	700 / 640	20.2 / 23.7	42.9 / 40.1	44.7 / 55.7	20.7 / 25.6	7.1 / 2.1	31.5 / 23.1

Table 4.1: Corpora statistics: number of parallel lines ($\times 10^3$) and proportion in the basic domain mixture (which does not include the NEWS domain), number of tokens in English and French ($\times 10^6$), number of types in English and French ($\times 10^3$), number of types that only appear in a given domain ($\times 10^3$). MED is the largest domain, containing almost 70% of the sentences, while REL is the smallest, with only 3% of the data.

	LAW	BANK	TALK	IT	REL
MED	1.93	1.97	1.9	1.93	1.97
LAW		1.94	1.97	1.93	1.99
BANK			1.98	1.94	1.99
TALK				1.92	1.93
IT					1.99

Table 4.2: The $\mathscr{H}$-divergence between domains

4.4.2 Baselines

Our baselines are standard for multi-domain systems.[9] Using Transformers implemented in OpenNMT-tf[10] (Klein et al., 2017), we build the following systems:

- a generic model trained on a concatenation of all corpora (Mixed). We develop two versions[11] of this system, one where the domain heterogeneity reflects the distribution of our train data given in Table 4.1 (Mixed-Nat) and one where all domains are equally represented in training (Mixed-Bal). The former is the best option when the train mixture $\mathscr{D}^s$ is also expected in testing; the latter should be used when the test distribution is uniform across domains. Accordingly, we report two aggregate scores: a weighted average reflecting the train distribution, and an unweighted average, meaning that test domains are equally important;
- fine-tuned models (Luong & Manning, 2015; Freitag & Al-Onaizan, 2016), based on the Mixed-Nat system. Further description of the implementation can be found in Appendix A. The full fine-tuning (FT-Full) procedure may update all the parameters of the initial generic model, resulting in six systems adapted for one domain, with no parameter sharing across domains.

[9] We however omit domain-specific systems trained only with the corresponding subset of the data, which are always inferior to the mix-domain strategy (Britz et al., 2017).

[10] https://github.com/OpenNMT/OpenNMT-tf

[11] In fact three: to enable a fair comparison with WDCMT, a RNN-based variant is also trained and evaluated. This system appears as Mixed-Nat-RNN in Table 4.3.

All models use embeddings and the hidden layers sizes of dimension 512. Transformers contain with 8 attention heads in each of the 6+6 layers; the inner feedforward layer contains 2048 cells. The adapter-based systems (see below) additionally use an adaptation block in each layer, composed of a 2-layer perceptron, with an inner ReLU activation function operating on normalized entries of dimension 1024. Training uses batches of 12,288 tokens, Adam with parameters $\beta_1 = 0.9$, $\beta_2 = 0.98$, Noam decay ($warmup\_steps = 4000$), and a dropout rate of 0.1 in all layers.

4.4.3 Multi-domain systems

Our comparison of multi-domain systems includes our own reimplementations of recent proposals from the literature:[12]

- a system using domain control as in Kobus et al. (2017): domain information is introduced either as an additional token for each source sentence (`DC-Tag`) or as a supplementary feature for each word (`DC-Feat`).
- a system using lexicalized domain representations presented in Chapter 5: sparse word embeddings are composed of domain-agnostic units and domain-specific units which will be nullified when translating in other domains (`LDR`);
- the three proposals of Britz et al. (2017). `TTM` is a feature-based approach where the domain tag is introduced as an extra word *on the target side*. The training uses reference tags, and inference is usually performed with predicted tags, just like for regular target words. `DM` is a multi-task learner where a domain classifier is trained on top of the MT encoder to make it aware of domain differences. `ADM` is the adversarial version of `DM`, pushing the encoder towards learning domain-independent source representations. These methods thus only use domain tags in training.
- the multi-domain model of Zeng et al. (2018) (`WDCMT`), where a domain-agnostic and a domain-specialized representation of the input are simultaneously processed; supervised classification and adversarial training are used to compute these representations. Again, inference does not use domain tags.[13]
- two multi-domain versions of the approach of Bapna & Firat (2019b), denoted `FT-Res` and `MT-Res`, which will be compared extensively in Chapter 6. The former variant corresponds to the original proposal of Bapna & Firat (2019b) (see also Sharaf et al. (2020)). It fine-tunes the adapter modules of a `Mixed-Nat` system independently for each domain, keeping all the other parameters frozen. The latter uses the same architecture but trains all parameters jointly from scratch with a

[12] Further implementation details are in Appendix A.

[13] For this system, we use the available RNN-based system from the authors (`https://github.com/DeepLearnXMU/WDCNMT`) which does not directly compare to the other, Transformer-based, systems; the improved version of Su et al. (2019) seems to produce comparable, albeit slightly improved, results.

mix-domain corpus as in multi-task training (Caruana, 1997). The loss function of MT-Res is the same as in Equation 5.1.

This list includes systems that slightly depart from our definition of MDMT. Standard implementations of TTM and WDCMT rely on the domain inference, rather than on gold, domain tags - which must somewhat affect their predictions. DM and ADM make no use of domain tags at all.

4.5 Results and discussion

4.5.1 Computational costs

The strategy "one domain / one model" allocates 65m parameters for one domain. The second most expensive method is to use residual adapters, including FT-Res and MT-Res, which use 12.4 additional parameters per domain. LDR follows residual adapters using $4 \times |\Sigma_{e,f}| \times 3$ ($\sim$384k) additional parameters per domain ($\Sigma_{e,f}$ is the joint vocabulary of the source and target languages). Domain tags such as TTM, DC-Tag spend only 512 additional parameters, which account for one additional embedding, for one domain. Domain embeddings DC-Feat add only 4 parameters per domain to the MDMT system. WDCMT's special gates consist of few thousand additional parameters which are however shared for every domain. Other methods such as DM, ADM do not use any additional parameters.

While FT-Res finetunes residual adapters during at most 50k iterations, the other methods consume the same number of training iterations as generic models. The training and inference latency is proportional to the number of parameters in each MDMT model. Indeed, MT-Res has largest latency while other MDMT systems perform as fast as generic models.

4.5.2 Performance of MDMT systems

In this section, we discuss the basic performance of MDMT systems trained and tested on 6 domains. Results are in Table 4.3. There is a large difference between Mixed-Nat and Mixed-Bal. The former system trained with balancing data in the generic setting is 2 BLEU points better in the uniform average, notably owing to the much better results for REL. As explained above, this setting should be the baseline when the test distribution is assumed to be balanced across domains. We use the weighted average to perform global comparisons as all other MDMT systems are trained with unbalanced data distribution.

Fine-tuning each domain separately yields a better baseline, outperforming Mixed-Nat for all domains, with significant gains for domains that are distant from MED: REL, IT, BANK, LAW.

Model / Domain		MED	LAW	BANK	TALK	IT	REL	WAVG	AVG
Mixed-Nat	[65m]	37.3	54.6	50.1	33.5	43.2	77.5	41.1	49.4
Mixed-Bal	[65m]	35.3	54.1	52.5	31.9	44.9	89.5	40.3	51.4
FT-Full	[6×65m]	37.7	**59.2**	**54.5**	34.0	**46.8**	**90.8**	**42.7**	**53.8**
DC-Tag	[+6×512]	38.1	55.3	49.9	33.2	43.5	**80.5**	41.6	50.1
DC-Feat	[+6×4]	37.7	54.9	49.5	32.9	43.6	**79.9**	41.4	49.9
LDR	[+6×384k]	37.0	54.7	49.9	33.9	43.6	**79.9**	40.9	49.8
TTM	[+6×1024]	37.3	54.9	49.5	32.9	43.6	**79.9**	41.0	49.7
DM	[+0]	35.6	49.5	45.6	29.9	37.1	62.4	38.1	43.4
ADM	[+0]	36.4	53.5	48.3	32.0	41.5	73.4	38.9	47.5
FT-Res	[+6×12.4m]	37.3	**57.9**	**53.9**	33.8	**46.7**	**90.2**	**42.3**	**53.3**
MT-Res	[+6×12.4m]	37.9	**56.0**	**51.2**	33.5	44.4	**88.3**	42.0	**51.9**
Mixed-Nat-RNN	[51m]	36.8	53.8	47.2	30.0	35.7	60.2	39.2	44.0
WDCMT	[73m]	36.0	53.3	**48.8**	31.1	**38.8**	58.5	39.0	44.4

Table 4.3: Translation performance of MDMT systems based on the same Transformer (top) or RNN (bottom) architecture. The former contains 65m parameters, the latter has 51m. For each system, we report the number of additional domain specific parameters, BLEU scores for each domain, domain-weighted (WAVG) and unweighted (AVG) averages. For weighted-averages, we take the domain proportions from Table 4.1. Boldface denotes significant gains with respect to Mix-Nat (or Mix-Nat-RNN, for WDCMT), underline denotes significant losses.

All MDMT systems (except DM and ADM) slightly improve over Mixed-Nat(for most domains), but these gains are rarely significant. Among systems using an extra domain feature, DC-Tag has a small edge over DC-Feat and also requires less parameters; it also outperforms TTM, which however uses predicted rather than gold domain tags in the inference. TTM is also the best choice among the systems that do not use domain tags in inference. The best MDMT candidates overall are FT-Res and MT-Res, which significantly improve over Mixed-Nat for a majority of domains, and are the only ones to clearly fulfill [P1-LAB]; WDCMT also improves on three domains, but regresses on one. The use of a dedicated adaptation module thus seems better than feature-based strategies, but yields a large increase of the number of parameters. The effect of the adaptation layer is especially significant for small domains (BANK, IT and REL).

All systems fail to outperform fine-tuning, sometimes by a wide margin, especially for an "isolated" domain like REL. This might be due to the fact that domains are well separated (cf. Section 4.4.1) and are hardly helping each other. In other words, the statistical bias is not well compensated by the variance reduction. In this situation, MDMT systems should dedicate a sufficient number of parameters to each domain, so as to close the gap with fine-tuning.

Set-up	Split		Split		Split		Merge		Wrong	
Model	MED (0.5 / 0.5)		MED (0.25 / 0.75)		LAW (0.5 / 0.5)		BANK+LAW		RND	NEW
	MED_1	MED_2	MED_1	MED_2	LAW_1	LAW_2	BANK	LAW	ALL	NEWS
FT-Full	-0.1	-0.6	-1.5	-0.2	-2.3	-5.1	-1.6	-1.4	-19.6	-3.3
DC-Tag	-0.2	-0.3	**+0.1**	+0.2	-0.4	-0.4	-0.5	-0.4	-13.4	-1.7
DC-Feat	-0.5	0.0	**+0.3**	+0.3	+0.3	+0.3	+0.3	+0.1	-14.2	-1.8
LDR	+0.1	+0.1	+0.4	+0.4	0.0	0.0	0.0	+0.1	-12.0	-1.4
TTM (*)	-0.2	-0.2	-0.2	-0.2	-0.3	-0.3	0.0	-0.3	0.0	-0.1
DM (*)	-0.3	-0.3	+0.4	+0.4	+0.3	+0.3	+0.9	+0.1	0.0	-0.9
ADM (*)	+0.6	+0.6	+0.4	+0.4	+0.4	+0.4	+0.1	-0.4	0.0	-0.2
FT-Res	-0.1	-0.4	-0.3	-0.3	-2.2	-2.9	-2.4	-3.2	-13.3	-3.0
MT-Res	-0.2	-0.1	**+0.2**	+0.0	-0.9	-0.9	+0.7	-0.3	-18.6	-1.3
WDCMT (*)	-0.0	-0.0	+0.2	+0.2	+0.8	+0.8	-0.4	-0.8	0.0	+0.2

Table 4.4: Translation performance with variable domain definitions. In the *Split*/*Merge* experiments, we report BLEU differences for the related test set(s). In the test NEW, we report BLEU differences between each MDMT system and the generic system Mixed-Nat. In the test RND, we report BLEU differences between using random domain tag and using true domain tag. Underline denotes significant loss when domains are changed wrt. the baseline situation; bold for a significant improvement over FT-Full; (*) MDMT systems ignoring test domains.

4.5.3 Redefining domains

Table 4.4 summarizes the results of four experiments where we artificially redefine the boundaries of domains, to challenge requirements [P2-TUN], [P3-HET], and [P4-ERR]. In the first three, we randomly *split* one corpus in two parts and proceed as if this corresponded to two actual domains. An MD system should detect that these two pseudo-domains are mutually beneficial and should hardly be affected by this change with respect to the baseline scenario (no split). In this situation, we expect MDMT to even surpass fine-tuning separately on each of these dummy domains, as MDMT exploits all data. In contrast, the fine-tuning method focuses only on a subpart. In testing, we decode the test set twice, once with each pseudo-domain tag. This makes no difference for TTM, DM, ADM and WDCMT, which do not use domain tags in testing. In the *Merge* experiment, we merge two corpora in training to assess the robustness to heterogeneous domains [P3-HET]. We then translate the two corresponding tests with the same (merged) system.

Our findings can be summarized as follows. For the *Split* experiments, we see small variations that can be positive or negative compared to the baseline situation, but these are hardly significant. All systems show some robustness with respect to fuzzy domain boundaries; this is mostly notable for ADM, suggesting that when domains are close, ignoring domain differences is effective. In contrary, FT-Full incurs clear losses across the board, especially for the small data condition (Miceli Barone et al., 2017). Even in

this very favorable case however, very few MDMT systems are able to significantly outperform FT-Full and this is only observed for the smaller part of the MED domain. The *Merge* condition is hardly different, with again large losses for FT-full and FT-Res, and small variations for all systems. We even observe some rare improvements with respect to the situation where we use actual domains.

4.5.3.1 Handling wrong or unknown domains

In the last two columns of Table 4.4, we report the drop in performance when the domain information is not correct. In the first (RND), we use test data from the domains seen in training, presented with a random domain tag. In this situation, the loss with respect to using the correct tag is generally large (more than 10 BLEU points), showing an overall failure to meet requirement [P4-ERR], except for systems that ignore domain tags in testing.

In the second (NEW), we assess [P5-UNK] by translating sentences from a domain unseen in training (NEWS). For each sentence, we automatically predict the domain tag and use it for decoding.[14] In this configuration, again, systems using domain tags during inference perform poorly, significantly worse than the Mixed-Nat baseline (Bleu=23.5).

4.5.3.2 Handling growing numbers of domains

Another set of experiments evaluate the ability to dynamically handle supplementary domains (requirement [P6-DYN]) as follows. Starting with the existing MD systems of Section 4.5.2, we introduce an extra domain (NEWS) and resume training with this new mixture of data[15] for 50,000 additional iterations. We contrast this approach with training all systems from scratch and report differences in performance in Figure 4.1 (see also Table B.1 in Appendix B).[16]

We expect that MDMT systems should not be too significantly impacted by adding a new domain and reach about the same performance as when training with this domain from scratch. From a practical viewpoint, dynamically integrating new domains is straightforward for DC-Tag or TTM, for which new domains merely add new labels. DC-Feat, MT-Res and FT-Res can also easily add new domain embeddings and residual

[14] Domain tags are assigned as follows: we train a language model for each domain and assign a tag to a sentence basis based on the language model log-probability (assuming uniform domain priors). This domain classifier has an average prediction error of 16.4% for in-domain data.

[15] The design of a proper balance between domains in training is critical for achieving optimal performance: as our goal is to evaluate all systems in the same conditions, we consider a basic mixing policy based on the new train distribution.

[16] WDCMT results are excluded from this table, as resuming training proved difficult to implement.

adapters respectively. It is less trivial for DM, ADM and WDCMT, which include a built-in domain classifier whose outputs have to be pre-specified or for LDR where domain-specific units have to be predefined. This makes a difference between domain-bounded systems, for which the number of domains is limited, and truly domain-open systems.

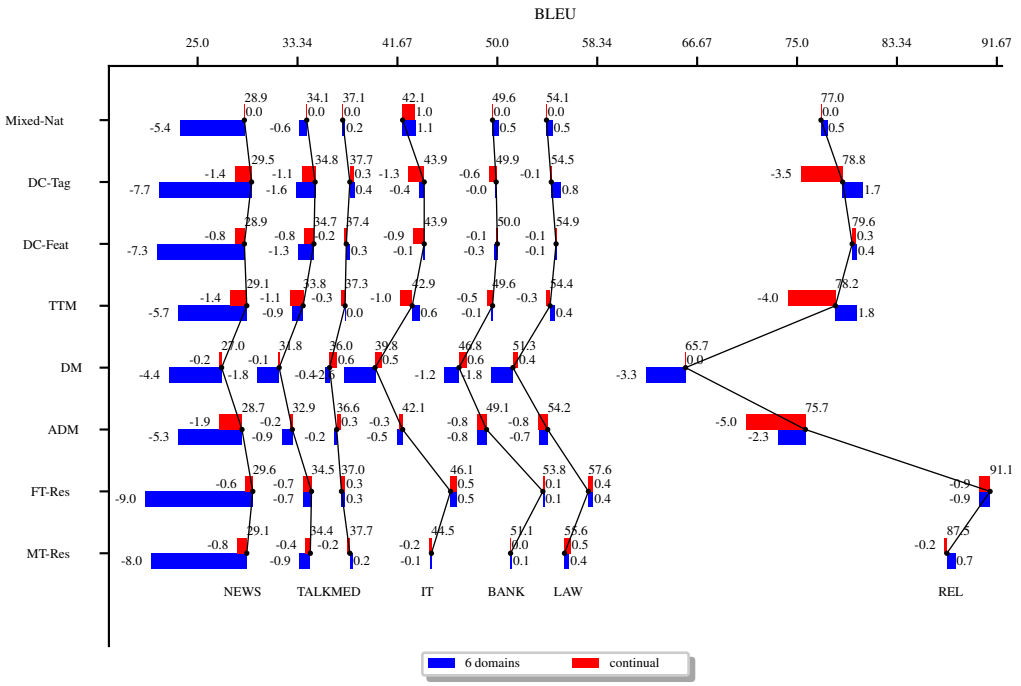


Figure 4.1: Ability to handle a new domain. We report BLEU scores for a complete training session with 7 domains, as well as differences (in blue) with training with 6 domains (from Table 4.3); and (in red) differences with continual training

.

First, we compare the results of coldstart training with six and seven domains in Table B.1. We observe that the extra train data is hardly helping for most domains, except for NEWS, where we see a significant gain, and for TALK. The picture is the same when one looks at MDMTs, where only the weakest systems (DM, ADM) seem to benefit from more (out-of-domain) data.

Secondly, we compare the coldstart with the warmstart scenario. We see that the former is always significantly better for NEWS, as expected, and that resuming training also negatively impacts the performance for other domains. This happens notably for DC-Tag, TTM and ADM. In this setting, MT-Res and DM show small average losses, with the former achieving the best balance of training cost and average BLEU score.

Model/ Clusters	Train size	Mixed Nat	FT Full	FT Res	MT Res	DC Feat	DC Tag	TTM	ADM	DM	LDR
10 small	29.3k	68.3	70.0	70.7	**71.2**	70.6	53.1	67.3	69.8	67.0	70.2
10 mid	104.7k	44.8	**48.0**	46.0	45.7	44.8	44.3	44.5	43.7	41.6	44.5
10 large	251.1k	50.4	**52.9**	52.0	51.3	49.6	43.2	49.1	48.5	44.3	49.5
Avg	128.4k	54.5	**57.0**	56.2	56.1	55.0	46.9	53.6	54.0	51.0	54.7

Table 4.5: BLEU scores computed by merging the 10 smaller, medium, and larger cluster test sets. Best score for each group is in boldface. For the small clusters, full-fine tuning is outperformed by several MDMT systems - see details in Appendix C.

4.5.4 Automatic domains

In this section, we want to evaluate the performance of MDMT systems in a large-scale setting that consists of a high number of domains. Without access to an actual large condition, we propose an artificial setting with automatic domains, obtained by clustering sentences of the mixed corpus into $k = 30$ classes using the k-means algorithm based on generic sentence representations obtained via mean pooling (cf. Section 4.4.1). This allows us to evaluate the requirement [P7-scale]. We train and test our MDMT systems as if these clusters were distinct domains. Many of these clusters are mere splits of the large MED, while a fewer number of classes are mixtures of two (or more) existing domains (full details are in Appendix C). We are thus in a position to reiterate, at a larger scale, the measurements of Section 4.5.3 and test whether multi-domain systems can effectively take advantage of the cross-domain similarities and perform better than fine-tuning eventually. The results in Table 4.5 also suggest that MDMT can surpass fine-tuning for the smaller clusters; for the large clusters, this is no longer true. The complete table (in Appendix C) shows that this effect is more visible for small subsets of the medical domain.

Finally, Table 4.6 reports the effect of using automatic class index instead of true domain. For each of the 6 test sets, each sentence was first assigned to an automatic class, then translated with the corresponding multi-domain system with 30 classes. We compare the performance of MDMT systems on 30-automatic-class scenario with ones in the 6-domain scenario. Results are clear and confirm previous observations: even though some clusters are very close, the net effect is a loss in performance for almost all systems and conditions.

4.6 Conclusions and outlook

In this study, we have carefully reconsidered the idea of multi-domain machine translation, which seems to be taken for granted in many recent studies. We spelled out the various motivations for building such systems and the associated expectations in terms of system performance. We designed a series of requirements that MDMT systems should

Domain / Model	MED	LAW	BANK	TALK	IT	REL	WAVG	AVG
DC-Tag	38.5	54.0	49.0	33.6	42.2	76.7	41.6	49.0
DC-Feat	37.3	54.2	49.3	33.6	41.9	75.8	40.8	48.7
LDR	37.4	54.1	48.7	32.5	41.4	75.9	39.1	48.3
TTM	37.4	53.7	48.9	32.8	41.3	75.8	40.7	48.3
DM	35.4	49.3	45.2	29.7	37.1	60.0	37.8	42.8
ADM	36.1	53.5	48.0	32.0	41.1	72.1	39.5	47.1
FT-Res	37.5	55.7	51.1	33.1	44.1	86.7	41.6	51.4
MT-Res	37.3	55.5	50.2	32.2	42.1	86.7	41.2	50.7
WDCMT	35.6	53.1	48.4	30.5	37.7	56.0	38.5	43.6

Table 4.6: Translation performance with automatic domains, computed with the original test sets. Significance tests are for comparisons with the 6-domain scenario (Table 4.3).

meet and proposed a series of associated test procedures. In our experiments with a wide range of MDMT methods, we have found that most requirements were hardly met for our experimental conditions. Effectively, when MDMT systems outperform the mixed-domain baseline, at least for some domains, they all fall short of matching the performance of fine-tuning on each domain, which remains the best choice in single domain adaptation.

However, MDMTs are less brittle than fine-tuning when domain frontiers are uncertain and can, to a certain extend, dynamically accommodate additional domains, this being especially easy for feature-based approaches. Our experiments finally suggest that all methods show decreasing performance when the number of domains or the diversity of the domain mixture increases.

Two other main conclusions can be drawn from this study. First, it seems that more work is needed to make MDMT systems make the best out of the variety of the available data, both to effectively share what needs to be shared while at the same time separating what needs to be kept separated. Second, and maybe more importantly, there is a general need to adopt better evaluation methodologies for evaluating MDMT systems. Systems developers should spell out the test conditions and the associated distribution of test instances and use as many domains as possible since a great variety of data is available nowadays.

Chapter 5

Lexicalized domain representations and Contextualized domain representation

5.1 Introduction

In this chapter, we present an MDMT system that we developed and published in Pham et al. (2019). Our system "Lexicalized domain representation" (LDR) was introduced in Section 4.4.3 of the previous chapter and used as an MDMT baseline in our overview. Recently, we extended the idea of LDR, in which a number of units in the 0^{th} layer are reserved for one domain and are dropped when translating in other domains, to Contextualized domain representations (CDR), in which not only the 0^{th} layer but also higher layers use the same dropping mechanism.

The *supervised multi-domain adaptation* setting presented in Section3.3 usually involves a large number of domains. Adapting the whole NMT system separately to each domain would waste training resources and require large hardware resources for maintenance. We develop a simple MDMT system that changes only the form of word embeddings, i.e., can be applied to any NMT architecture. Our proposal transposes to neural machine translation the feature expansion technique of Daumé III (2007). We isolate domain-agnostic from domain-specific units in word embeddings while sharing the rest of the network across domains. Our main hypothesis is that domains mostly differ at the lexical level due to cross-domain polysemy, which motivates domain-specific embeddings. Another motivation to reconstruct only word embeddings is that we can easily handle the growing number of domains by creating more domain-specific units, which can not be done so easily in higher layers.

CDRs are extensions of LDRs in higher levels of an NMT model. These layers create a partition of nodes in the network of an NMT model assigning different sub-networks to different domains. By doing this, we reduce the interference between domains in the contribution of the parameters to the prediction. Our first version of CDRs is hard-coded, i.e., the choice of dropping mask is heuristically predefined. However, the dropping pattern

can be learned via a latent variable Gong et al. (2021b,a).

Our experiments are complementary to those in Chapter 4 and consider four domains, two neural architectures, and two language pairs and find that our technique yields effective multi-domain NMTs, outperforming several baselines. We also report the performance of LDR in a highly heterogeneous MDMT setting (see Chapter 4). We evaluate CDRs in the same MDMT setting as in Chapter 4.

Our contributions are thus as follows: we adapt and implement the ideas of Daumé III (2007) for two NMT architectures; we provide experimental evidence that shows the effectiveness of this technique; we evaluate the ability of our networks to accommodate new domains dynamically; we apply the idea of the sparse representation from LDR to higher layers and receive promising results in a highly heterogeneous MDMT setting. Finally, we introduce a new technique to analyze word polysemy using embeddings, which comforts the assumption that their variation across domains reflects the change of senses.

This chapter draws from the following publication: Pham et al. (2019).

5.2 Lexicalized domain representations

5.2.1 Multi-domain machine translation

As introduced in Section 3.2.2, a train example in the *supervised multi-domain adaptation* setting is a triplet (d,x,y), with x in the source language, y in the target language and d a domain tag in $[1 \dots n_d]$. According to Section 3.2.1 the training instances are distributed according to a mixture $\mathscr{D}_e^S$ such that $\mathscr{D}_e^S(x) = \sum_{d=1}^{n_d} \lambda^s(d) \mathscr{D}_e^d(x)$, with $\{\lambda^s(d), d = 1 \dots n_d\}$ the mixture weights satisfying $\sum_d \lambda^s(d) = 1$. According to our definition of the NMT loss function in Equation (2.25) and our definition of multi-domain distribution in Section 3.2.1 and Section 3.1, our objective is to find a tuple of parameters $\{\theta_1 \dots \theta_d\} \in \mathbb{R}^D \times \dots \times \mathbb{R}^D$ minimizing:

$$\sum_{d=1}^{n_d} \lambda^s(d) E_{x \sim \mathscr{D}_e^d, y \sim g^d(x)}[-log(P(y|x,\theta_d))]. \tag{5.1}$$

in which the labeling function of each domain d will be $g^d : \Omega_e \rightarrow \Omega_f$ where Ω_e and Ω_f are the set of sentences in the language e and the language f respectively.

A straightforward solution is to process each domain separately, computing the value θ_d^* that minimizes the empirical loss in $\mathscr{D}_e^d$ and g^d. This strategy is only effective if we have sufficient training data for each domain; when this is not the case, some estimates θ_d^* may be far from their optimal value. The alternative we consider here constraints each parameter θ_d to be made of two parts: $\theta_d = [\theta_s; \theta'_d]$. $\theta_s \in R^{D_g}$ is shared across all domains,

while the second part $\theta'_d \in \mathbb{R}^{D_d}$ is only used in domain d. The parameter set is much more constrained, yet we expect that tying parameters across domains will yield better estimates for θ_s due to a larger training corpus. In this setting, the optimization program defined by equation (5.1) can no longer be performed separately for each training corpus.

5.2.2 Lexicalized domain embeddings

To actually implement this idea for NMT, we need to define the subset of parameters that will be shared across domains. In this section, we explore the hypothesis that domain specificities can be confined to the lexical level, and we define θ_s to contain all the network parameters except for a subpart of the word embeddings. For each word v, the embedding vector $e(v)$ is thus decomposed as $e(v) = [e_g(v); e_1(v); \ldots; e_d(v)]$, where $e_g(v)$ stores the domain-agnostic lexical embedding, while $e_d(v)$ stores the subpart that is specific to domain d. In our NMT architectures, the actual embedding layer composes these vectors linearly to generate the word embedding for domain k according to:

$$\begin{aligned}\tilde{e}_k(v) =& M_g e_g(v) + \sum_{d \in [1,..,n_d]} M_d \times e_d(v) \times \delta(d = k) \\ =& M[e_g(v); e'_1(v,k) \ldots; e'_d(v,k)],\end{aligned} \tag{5.2}$$

where $\delta()$ is the indicator function, M is the matrix made of blocks $M_g, M_1 \ldots, M_{n_d}$, and $e'_d(v,k)$ is the masked embedding: $e'_d(v,k) = e_d(v) * \delta(d = k)$.

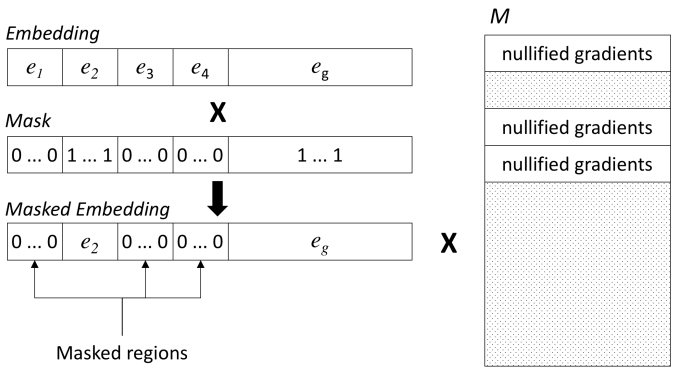


Figure 5.1: Lexicalized domain embeddings. When processing a sample from domain 2, we only activate the corresponding parameter region (θ_2) in the input embeddings; the remaining domain-specific parts are zeroed out and do not receive any update. The domain-agnostic part is always active and is updated irrespective of the input domain.

Ensuring that the actual embedding does not contain any zero is essential for the Transformer model since the lexical representations are added to the positional encoding, which would undo the effect of domain masking and propagate a gradient even to regions that

should not be modified. With our design, we make sure that the matrix M receives gradient 0 at regions corresponding to deactivated regions in the word embedding during backpropagation. Those regions are also masked in the forward step. Thus they do not interfere with the training on the domains to which they are not assigned (see Figure 5.1). Our architecture is thus readily compatible with any NMT architecture, where we replace standard embedding layers with the embeddings defined in equation (5.2). In our experiments, we consider both the attentional RNN architecture of Bahdanau et al. (2015) and the Transformer architecture of Vaswani et al. (2017).

5.3 Contextualized domain representations

CDRs replace the linear combination of domain-agnostic units and domain-specific units used to compute LDR embeddings by multiplying the output of each layer with domain-specific dropping vector as follows

$$\begin{aligned} \tilde{h}^l &= h^l * r^l(d) \\ r^l(d) &\in \mathbb{R}^{d_k} \\ r^l(d)_i &= \begin{cases} 1, & \text{if } i < d_a \\ 1, & \text{if } d_a + d \times \frac{d_k - d_a}{n_d} \leq i < d_a + (d+1) \times \frac{d_k - d_a}{n_d} \\ 0, & \text{otherwise} \end{cases} \\ d &\in \{0, \ldots, n_d - 1\} \end{aligned} \tag{5.3}$$

h_l is the output of the l^{th} layer; d_a is the number of domain-agnostic nodes; n_d is the number of domains. We allocate the same number of domain-specific nodes for every domain. However, the number of nodes in each intermediate layer is always fixed at d_k, CDR can not dynamically create more units for new domains. Therefore, the model currently only handles a fixed number of domains. In Section 5.6 we will discuss a promising approach that allows CDR to handle an arbitrary number of domains.

In our experiments with CDR, we consider only the Transformer architecture and the same experimental setting as in Section 4.4.1.

5.4 Experiments

5.4.1 Domains, data and metrics

5.4.1.1 4 domains, 2 language pairs

We experiment with two language pairs (English-French, English-German) and data originating from three domains, corresponding to texts from three European institutions: the

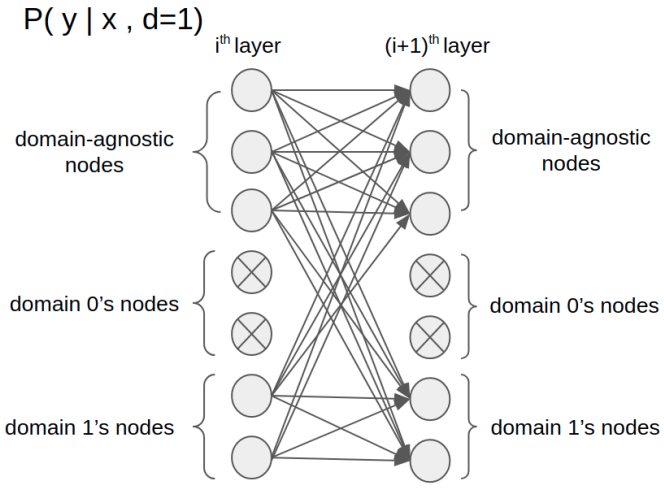


Figure 5.2: Contextualized domain representations. When processing a sample from domain 1, only the signal of the domain-agnostic nodes and the 1^{th} domain's nodes are passed to higher layers, other nodes are dropped out.

European Parlement (EPPS) (Koehn, 2005), the European Medicines Agency (EMEA), the European Central Bank (ECB) (Tiedemann, 2009), In addition, for English-French we also use corpora in the IT-domain obtained from the OPUS web site[1] corresponding to KDE, Ubuntu, GNOME and PHP datasets (IT). Moreover, we use IT only in the scenario of continual training, in which we continue training our MDMT system with the mix of new domain IT and the old domains EMEA, ECB and EPPS.

We randomly split those corpora into training, validation and test sets (see statistics in Table 5.1). Validation sets are used to chose the best model according to the average BLEU score (Papineni et al., 2002)[2]

Corpus	Train	Valid	Test
English → French			
EMEA	1.09M	1,000	$1{,}000_{(300)}$
ECB	0.19M	1,000	1,000
EPPS	2.01M	1,000	1,000
IT	0.54M	1,000	1,000
English → German			
EMEA	1.11M	1,000	$1{,}000_{(300)}$
ECB	0.11M	1,000	1,000
EPPS	1.92M	1,000	1,000

Table 5.1: Corpora statistics.

Note that the EMEA dataset distributed on the OPUS site contains multiple sentence

[1] `http://opus.nlpl.eu`
[2] We use detokenized, truecasing multibleu.

duplicates. We therefore report below two numbers as $S_{(T)}$: the first (S) is comparable to what has been published on earlier studies (e.g. Zeng et al. (2018)), the second one (T) is obtained by making the test entirely disjoint from the training (700 duplicated sentences are discarded).

To reduce the number of lexical units and make our systems open-vocabulary, we apply Byte-Pair Encoding (Sennrich et al., 2016b) separately for each language with 30,000 merge operations.

5.4.1.2 7 domains, 1 language pair

We report again the performance of `LDR` in the same setting presented in Section 4.4.1, which consists of 7 domains in English→French translation task. For the sake of brevity, we will not provide the description of the data here (see Section 4.4.1).

5.4.1.3 Statistical significance

Statistical significance is estimated using bootstrap resampling (Koehn, 2004), implemented in compare-mt[3] (Neubig et al., 2019). We report significant differences at the level of $p = 0.05$.

5.4.2 Baselines

5.4.2.1 3 domains, 2 language pairs

To validate our findings, we compared lexicalized domain embedding models with standard models using both attentional Recurrent Neural Networks (RNNs) and the Transformer architecture. Our baselines consist of:

- generic models trained with a simple concatenation of all corpora (`Mixed-Nat`);
- models tuned separately on each domain for respectively (10000, 15000, 5000) iterations using in-domain data ($\texttt{ft}_{EMEA}$, $\texttt{ft}_{EPPS}$, $\texttt{ft}_{ECB}$);
- models using domain tags as in Kobus et al. (2017) (`DC`);

For all models, we set the embeddings size equal to 512; the size of hidden layers is equal to 1024 for RNNs and 512 for Transformer. Other important configuration details are as follows: Transformer models use multi-head attention with 8 heads in each of the 6 layers; the inner feedforward layer contains 2048 cells; RNN models use 1 layer on both sides: a bidirectional LSTM encoder and a unidirectional LSTM decoder with attention. The domain control systems are exactly as their baseline counterparts (RNN and Transformer), with an additional 2 cells encoding the domain on the input layer. To

[3] `https://github.com/neulab/compare-mt`

train NMT systems, we use Adam, with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\alpha = 0.0005$ for RNNs; with parameters $\beta_1 = 0.9$, $\beta_2 = 0.98$, with Noam decay (Vaswani et al., 2017) for Transformer ($warmup\_steps = 4000$). In all cases, we use a batch size of 128 and a dropout rate of 0.1 for all layers. All our systems are implemented in OpenNMT-tf[4] (Klein et al., 2017).

5.4.2.2 6 domains, 1 language pair

The description of MDMT systems for the comparison with CDR and LDR was provided in Section 4.4.2 and in Section 4.4.3.

5.4.3 Implementing Lexicalized Domain Representations

5.4.3.1 3 domains, 2 language pairs

In order to implement LDR, we split the embedding vector into four regions: 3 are domain specific and 1 is domain-agnostic, with sizes $[8,8,8,488]$ respectively. If a sentence originates from domain i, the domain specific regions for all domains $j \neq i$ will be zeroed out while the other regions are activated (cf. Figure 5.1). We then use a dense layer of size 512 to fuse the region for the active domain and the domain-agnostic region. Training is formalised in algorithm 1. Note that each iteration of algorithm 1 uses 2 batches: a "generic" batch updating only the domain-agnostic region; and a "domain-specific" batch updating both the domain-agnostic and domain-specific parameters.

Algorithm 1 Multi-domain Training

Require: Corpora $C_i, i \in [1,..,d]$ for d domains, Batch size B

1: **repeat**
2: Randomly pick $i \in [1,..,d]$ w.r.t the multinomial distribution $[\frac{|C_i|}{\sum_{i \in [1,..,d]} |C_i|}]$.
3: Randomly pick B sentences from C_i.
4: Activate only domain-agnostic region to create generic batch, denoted W_g.
5: Compute gradient of θ_s, $\frac{\partial L}{\partial \theta_s}$ using W_g.
6: Activate domain-specific and domain-agnostic regions to create domain-specific batch W_i
7: Compute gradient of domain-specific parameters θ_i, $\frac{\partial L}{\partial \theta_i}$ using W_i.
8: Update parameters θ_s using $\frac{\partial L}{\partial \theta_s}(W_g)$ and θ_i using $\frac{\partial L}{\partial \theta_i}(W_i)$
9: **until** convergence

The batch selection procedure (step 2 of algorithm 1) ensures that the number of examples in each domain used in training follows the distribution of the training data, i.e., sentences from the Europarl domain will be selected more frequently that the two other

[4] https://github.com/OpenNMT/OpenNMT-tf

domains. We also consider a more balanced sampling procedure, where i is selected according to distribution $[\frac{\sqrt{|C_i|}}{\sum_{i\in[1,..,d]}\sqrt{|C_i|}}]$. The corresponding results are reported as $\texttt{LDR}^{0.5}$.

5.4.3.2 6 domains, 1 language pair

In this setting, we split the embedding vector of size 532 into seven regions: 6 are domain specific and 1 is domain-agnostic, with sizes $[4,4,4,4,4,4,508]$ respectively. By doing this, each domain will have 512 effective units. A dense layer maps the resulted sparse LDR embedding to an embedding of 512 units before passing to the encoder/decoder.

5.4.4 Implementing Contextualized domain representations

5.4.4.1 6 domains, 1 language pair

We do nothing special except applying domain dropping mask to each layer as explained in Equation (5.3) in both the encoder and the decoder of an NMT model. For the hyper-parameters, we choose heuristically $d_a = 480$ and $d_k = 672$ so that there are $\frac{672-480}{6} = 32$ domain-specific nodes for each domain, making the effective number of nodes for each domain equal to 512. This setting uses approximately $3m$ additional parameters per domains, which is only 25% compared to FT-Res and MT-Res (see Section 4.5.1 and Table 4.3).

In our experiments, we evaluate this method with Transformer models, which are state-of-the-art architecture in MT. We also report $\texttt{CDR}^{1.0}$ and $\texttt{CDR}^{0.5}$ which correspond to the distribution of simple concatenation of the domains' corpus and the down-sampling distribution $[\frac{\sqrt{|C_i|}}{\sum_{i\in[1,..,d]}\sqrt{|C_i|}}]$.

5.4.5 Results

5.4.5.1 3 domains, 2 language pairs

Results are summarized respectively in Table 5.2 for the Transformer systems and Table 5.3 for the RNN systems[5]. First, we observe that Transformers are consistently better than RNNs and that fine-tuning on a domain-specific corpus, when applicable, is almost the best way to optimize the performance on that domain.[6] Note that fine-tuning however yields a marked (even sometimes catastrophic, eg. for the EMEA-tuned Transformer system) decrease in performance for the other domains.

[5] As explained above, we report two numbers when testing with EMEA, except for the fine-tuning scenarios when tuning on ECB and EPPS.

[6] This is not so clear for EPPS, where fine-tuning does not seem to help.

Our approach (LDR$_{oracle}$) is consistently better than the Mixed-Nat strategy, with gains that range from very large (for EMEA and ECB) to unsignificant (for EPPS in most conditions). This means that our architecture is somehow able to compensate for the data unbalance and to raise the performance of the multi-domain system close to the best (fine-tuned) system in each domain. We even observe rare cases where the LDR$_{oracle}$ system outperforms fine-tuning (e.g. Transformer en:de in the EMEA domain). LDR$_{oracle}$ is also better than Domain Control in three conditions out of four, DC being seemingly a better choice for the RNN than for the Transformer architecture. As expected, ignoring the true domain label yields a light drop in performance: this is reflected in the results of LDR$_{pred}$, which relies on automatically predicted domain labels.[7] Note that this decrease is however hardly significant, showing that our architecture is quite robust to noisy labels. Even in the worst case scenario where all domain tags are intentionally wrong (LDR$_{wrong}$), we see that the domain-agnostic part still ensures a satisfying level of performance. A last contrast is with LDR$^{0.5}_{oracle}$ where we change the distribution of training sentences to decrease the weight of EPPS data and increase the number of ECB samples. As a result, we see a small decrease for EMEA and EPPS, and a large boost for ECB. This shows that our technique can be used in conjunction to other well known strategies for performing domain adaptation.

We also compare our architecture with the multi-domain model of Zeng et al. (2018) (WDCMT) for the pair English→French. We use the author's implementation[8] that is composed of one bidirectional Gated recurrent units (GRU) layer on the encoder side; and one unidirectional conditional GRU layer on the decoder side; the dimension of "domain" layers is 300. The direct comparison with our RNN is difficult, as both networks differ in many ways: framework, cell types, *etc*. Results in Table 5.4 therefore use a variant of our model that makes it more similar to the WDCMT network. In particular, this variant also uses a single GRU layer in the encoder and a single conditional GRU layer in the decoder (LDR$^{condgru}_{pred}$). As can be seen in this table, our model is on average comparable to WDCMT, while using a much simpler design.

5.4.5.2 6 domains, 1 language pair

To allow the readers easily compare the performance of LDR between 2 settings: 4 domains and 6 domains, we report again a part, which contains only Transformer-based system, of Table 4.3 (see Table 5.5).

In this setting, LDR is only slightly better than the generic model Mixed-Nat, which

[7] Our domain classifier uses a bi-LSTM RNN encoder, followed by a simple softmax layer. Its precision on a development set exceeds 95%.

[8] http://github.com/DeepLearnXMU/WDCNMT

Model	EMEA	EPPS	ECB	Avg.
English→French				
Mixed − Nat	$67.69_{47.60}$	37.50	53.49	52.89
FT_{EMEA}	$76.77_{49.43}$	17.16	11.99	35.30
FT_{EPPS}	20.86	37.04	24.53	27.47
FT_{ECB}	26.93	27.09	**56.52**	36.84
DC	$67.87_{45.42}$	37.31	54.14	53.10
LDR_{oracle}	$74.26_{\mathbf{49.90}}$	37.67	54.07	55.33
$\text{LDR}^{0.5}_{oracle}$	$\mathbf{74.95}_{49.38}$	37.35	55.91	**56.07**
LDR_{pred}	$74.29_{49.84}$	**37.73**	54.01	55.34
LDR_{wrong}	$72.95_{49.78}$	37.62	53.35	54.64
English→German				
Mixed − Nat	$64.57_{42.99}$	26.47	68.67	53.23
FT_{EMEA}	$68.35_{42.97}$	17.02	32.87	39.41
FT_{EPPS}	36.19	26.29	40.71	34.39
FT_{ECB}	24.72	18.36	**74.05**	39.04
DC	$63.48_{42.98}$	26.27	66.95	52.23
LDR_{oracle}	$70.90_{\mathbf{46.12}}$	26.30	68.90	55.36
$\text{LDR}^{0.5}_{oracle}$	$\mathbf{71.31}_{45.23}$	25.98	73.74	**57.01**
LDR_{pred}	$70.89_{\mathbf{46.12}}$	**26.53**	68.63	55.35
LDR_{wrong}	$69.51_{43.50}$	26.31	66.86	54.22

Table 5.2: BLEU scores for Transformer systems . Boldface denotes significant gains with respect to Mixed − Nat.

is trained with the same sampling distribution. The system is only significantly better than Mixed-Nat in REL. Moreover, LDR achieves a performance equivalent to DC-Tag, DC-Feat. This result is inconsistent with the results in the previous setting with only four domains. This might be explained by the interference between domains in the contribution of parameters in the prediction. As there are more domains and the domains' sizes are much more unbalanced, the partition of the parameters should be more extensive than the word embedding level. Moreover, word-embeddings do not capture the context, which is more important for the prediction than the word alone. We find that a partition of intermediate nodes in higher layers captures better the statistical bias than just in the 0^{th} layer, where the context is not formed. Our hypothesis is supported by the performance of $\text{CDR}^{1.0}$. Our model significantly outperforms Mixed-Nat in $4/6$ domains except for TALK and MED which are "hard" domains where no MDMT systems significantly does better than Mixed-Nat. Moreover, $\text{CDR}^{1.0}$ performs equivalently to MT-Res using 75% less additional parameters. Furthermore, with a sampling strategy more balanced, $\text{CDR}^{0.5}$ matches the performance of FT-Res, which is carefully fine-tuned for each domain.

However, Table 4.4 shows a dramatic loss in the random domain tag setting, which

Model	EMEA	EPPS	ECB	Avg.
English→French				
Mixed − Nat	$65.42_{45.11}$	34.70	51.38	50.50
$\texttt{FT}_{EMEA}$	$72.06_{\mathbf{47.33}}$	18.62	16.78	35.82
$\texttt{FT}_{EPPS}$	35.47	34.61	39.56	36.55
$\texttt{FT}_{ECB}$	21.93	22.60	51.53	32.02
DC	$68.26_{43.76}$	35.13	50.09	51.16
$\texttt{LDR}_{oracle}$	$71.73_{46.30}$	**35.21**	50.91	52.62
$\texttt{LDR}^{0.5}_{oracle}$	$71.70_{46.41}$	34.24	**52.37**	**52.77**
$\texttt{LDR}_{pred}$	$\mathbf{72.76}_{46.35}$	35.10	50.38	52.75
$\texttt{LDR}_{wrong}$	$62.10_{43.29}$	34.17	48.79	48.35
English→German				
Mixed − Nat	$57.37_{37.94}$	23.10	63.54	48.00
$\texttt{FT}_{EMEA}$	$65.64_{\mathbf{44.71}}$	12.36	15.93	31.31
$\texttt{FT}_{EPPS}$	24.90	22.98	26.26	24.71
$\texttt{FT}_{ECB}$	41.80	15.97	71.07	42.95
DC	$62.53_{39.25}$	**23.74**	65.71	50.66
$\texttt{LDR}_{oracle}$	$\mathbf{63.43}_{40.04}$	22.66	64.40	50.16
$\texttt{LDR}^{0.5}_{oracle}$	$63.27_{38.16}$	21.83	**69.55**	**51.55**
$\texttt{LDR}_{pred}$	$63.17_{39.92}$	22.51	64.00	49.89
$\texttt{LDR}_{wrong}$	$56.84_{37.05}$	22.06	61.66	46.85

Table 5.3: BLEU scores for RNN systems

Model	EMEA	EPPS	ECB	Avg.
English→French				
$\texttt{LDR}_{pred}$	$\mathbf{72.76}_{46.35}$	35.10	50.38	**52.75**
$\texttt{LDR}^{condgru}_{pred}$	$71.70_{46.21}$	35.09	51.22	52.67
WDCMT	$68.76_{45.29}$	**35.71**	**52.75**	52.40

Table 5.4: BLEU scores for RNN systems. Comparison between WDCMT and $\texttt{LDR}_{pred}$ built using conditional GRUs.

means LDR is not robust in the case of 6 domains. This observation is inconsistent with the results in Table 5.2. This loss might be due to the fact that this MDMT setting is much more adverse than the previous setting with only 3 domains. This again illustrates the need to use a wide range of available domains to evaluate the systems.

Model / Domain		MED	LAW	BANK	TALK	IT	REL	WAVG	AVG
Mixed-Nat	[65m]	37.3	54.6	50.1	33.5	43.2	77.5	41.1	49.4
Mixed-Bal	[65m]	35.3	54.1	52.5	31.9	44.9	89.5	40.3	51.4
FT-Full	[6×65m]	37.7	**59.2**	**54.5**	34.0	**46.8**	**90.8**	**42.7**	**53.8**
DC-Tag	[+6×512]	38.1	55.3	49.9	33.2	43.5	**80.5**	41.6	50.1
DC-Feat	[+6×4]	37.7	54.9	49.5	32.9	43.6	**79.9**	41.4	49.9
LDR	[+6×384k]	37.0	54.7	49.9	33.9	43.6	**79.9**	40.9	49.8
$CDR^{1.0}$	[+6×3m]	38.0	**55.9**	**51.4**	33.9	**45.0**	**87.4**	**42.4**	51.9
$CDR^{0.5}$	[+6×3m]	37.3	**57.0**	**54.5**	32.2	**47.6**	**91.0**	**42.4**	53.3
TTM	[+6×1024]	37.3	54.9	49.5	32.9	43.6	**79.9**	41.0	49.7
DM	[+0]	35.6	49.5	45.6	29.9	37.1	62.4	38.1	43.4
ADM	[+0]	36.4	53.5	48.3	32.0	41.5	73.4	38.9	47.5
FT-Res	[+6×12.4m]	37.3	**57.9**	**53.9**	33.8	**46.7**	**90.2**	**42.3**	**53.3**
MT-Res	[+6×12.4m]	37.9	**56.0**	**51.2**	33.5	44.4	**88.3**	42.0	**51.9**

Table 5.5: Translation performance of MDMT systems based on the same Transformer architecture. The former contains 65m parameters. For each system, we report the number of additional domain specific parameters, BLEU scores for each domain, domain-weighted (WAVG) and unweighted (AVG) averages. For weighted-averages, we take the domain proportions from Table 4.1. Boldface denotes significant gains with respect to Mix-Nat, underline denotes significant losses.

5.5 Complementary experiments

5.5.1 Balancing domain-agnostic and domain-specific representations in LDR

An important practical question concerns the balance between the domain-agnostic and the domain-specific part of the embeddings. In the limit where the domain specific part is very small, we should recover the performance of the Mixed-Nat system; conversely, we expect to see a less effective sharing of data across domains by increasing the domain-specific regions. Table 5.6 reports the result of a series of experiments for the Transformer architecture (English-French) with varying domain-specific sizes allocating between 4 and 64 cells for domain-specific information, and the complement to 512 for the domain-agnostic part. The differences are overall quite small in our experimental setting, where the training data is relatively limited and does not require to use a large embedding size. We therefore decided to allocate 8 cells for the domain specific part in the experiments of Section 5.4.3.1. This suggests that we could easily accommodate more domains with the same architecture and even reserve some regions to handle supplementary data (see below).

LDR$_{oracle}$	EMEA	EPPS	ECB	Avg.
English→French				
size=4	$74.65_{49.61}$	37.42	54.49	55.52
size=8	$74.26_{49.90}$	37.67	54.07	55.33
size=16	$74.15_{49.10}$	**37.78**	**54.56**	55.50
size=32	$\mathbf{75.10}_{48.61}$	37.64	54.29	**55.68**
size=64	$74.50_{\mathbf{50.17}}$	37.27	54.50	55.42

Table 5.6: BLEU scores for the Transformer architecture for varying domain-specific embedding sizes

5.5.2 Additional domain setting with LDR

5.5.2.1 4 domains

We now evaluate the ability of our model to integrate new domains, a very common scenario for industrial MT. In this setting, we consider that we have a model (LDR$_{oracle}$) trained as before for EMEA, EPPS and ECB during 200,000 iterations, which needs to process new training data from the IT domain. Assuming that we have reserved extra empty embedding cells[9] for this domain, we resume training with 4 domains during 100,000 additional iterations, yielding an updated model LDR$^{*}_{oracle}$. Results for the English→French language pair are in Table 5.7, where for comparison purposes we also report numbers obtained with continued training with the Mixed − Nat model, training for the same number of iterations and using the same four datasets (Mixed − Nat*).

Model	EMEA	EPPS	ECB	IT	Avg.
English→French					
Mixed − Nat	$67.69_{47.60}$	37.50	53.49	13.91	43.15
Mixed − Nat*	$66.49_{45.79}$	37.59	55.07	51.78	52.73
LDR$_{oracle}$	$74.26_{\mathbf{49.90}}$	**37.67**	54.07	13.40	44.85
LDR$^{*}_{oracle}$	$\mathbf{76.17}_{49.71}$	37.48	**55.12**	**55.24**	**56.00**

Table 5.7: BLEU scores for the Transformer architecture when including IT as additional domain

As expected, a huge improvement in performance is observed for the IT test set when learning includes in-domain data for both models, with LDR$^{*}_{oracle}$ outperforming Mixed − Nat* by a wide margin. It is interesting to see that this additional data has also a positive impact on other test sets: both models similarly increase their performance for the ECB domain, and LDR$^{*}_{oracle}$ additionally improves the results for the EMEA test, which is not the case

[9]For this experiment, word embeddings contain 480 cells for the domain-agnostic region and 32 cells for domain specific regions (8 cells x 4 regions).

for Mixed − Nat*; finally, using IT data does not impact the quality of translations for the EPPS domain of any of the models. Overall better results are obtained by our LDR^{*}_{oracle} model trained with data from an additional source.

5.5.2.2 7 domains

We report again the result of the experiments in the continuous learning presented in Section 4.5.3.2. We observe a significant loss of LDR compared to the generic baseline Mixed-Nat while adding domain NEWS. The loss is mainly due to a decrease of LDR's performance in the largest MED domain. This result shows the brittleness of our system against the change of train data distribution, in which the proportion of med drops 5% from 0.68 to 0.65.

5.5.3 Analysis of Word Embeddings of LDR

One of our main assumptions is that the difference between domains can be confined at the lexical level, warranting our decision to specialize lexical representations for each domain, while the remaining part of the network is shared across domains. Linguistically, this assumption relates to the classical "one sense per collocation" (Yarowsky, 1993) and corresponds to the fact that in many cases, polysemy corresponds to variation of use across domain. This variation of senses was reported in Carpuat et al. (2013) as a source of errors when translating in unseen domains. The authors demonstrated that a word would be translated to different translations with respect to the domains. SMT models can not boostrap this lack of knowledge and make sense errors. In MDMT, MT models cope the same problem. In the weaker form of (Yarowsky, 1993)'s theory, it allows us to assume that all occurrences of a given form in a given domain correspond to the same sense and share the same representation; the same form occurring in different domains is allowed to have one distinct embedding per domain, which may help capture polysemy and lexical ambiguity in translation.

To check this hypothesis, we performed the following analysis of embeddings learned with the multi-domain Transformer system for English:French. For each unit[10] in our English dictionary, we compute the k nearest neighbours for each domain $i \in [1 \dots d]$, where the distance between unit u and v for domain i is the cosine distance in the corresponding embedding space, ie. assuming that the actual embedding of v for domain i is $e(v,i) = M_g e_g(u) + M_i e_i(v)$ (cf. equation (5.2)). This process yields d lists of k near-

[10] In this study, we work with BPE units, in many cases we observe the variation of use of *word parts*. As we work with a large inventory, many of these units still correspond to actual words and we focus on these in our comments. We also restrict our analysis to words that occur at least 30 times in each domain, to ensure that each domain-specific region is updated during training.

est neighbours. A small intersection should then be a sign of a variation of use across domains; conversely, an near-identical set of neighbours across domains should reflect the stability of word use. Table 5.8 list the 10 units with the smaller (respectively larger) intersection (we use $k = 10$ and $d = 3$).

Polysemic "words"	Monosemic "words"
ases (0)	obtain (10)
impairment (1)	virtually (10)
convenience (1)	represent (10)
oring (1)	safety (10)
ums (1)	defence (10)
turnover (1)	coordinated (10)
occurrence (1)	handling (10)
tent (2)	July (10)
ture (2)	previous (10)
mation (2)	better (10)

Table 5.8: Analyzing the variation of embeddings across domains. For each word or subword we also report the size of the intersection (between 0 and 10).

Let us first consider the full words in the left column of Table 5.8. The case of *impairment* is pretty clear, occuring in EMEA mostly in terms such as "hepatic impairment" or "renal impairment", and translating into French as *insuffisance*. In ECB, its collocates are quite different and impairment often occurs in terms such as "cost subject to impairments" (French: *coût soumis à des réductions de valeur*). Likewise, "convenience" seems to have its general meaning ("for convenience") in EMEA, but appears in ECB in the specific context of "convenience credit card" (French: *carte de crédit à remboursement différé*). We finally see the same phenomena with "turnover", which is consistently translated with its economic meaning (French: *chiffre d'affaire*) in ECB and EPPS, but whose collocates in EMEA ("bone turnover", "hepatic turnover") are associated with the idea of the cell renewall process, yielding translations such as *remodelage osseux* in French. Subword units can be analysed in the same ways: "ums", for instance, appears in words such as "gums", "serums", "vacuums" in EMEA; in ECB, "ums" is mostly the suffix of "maximums", "minimums", or "premiums"; EPPS finally contains a more diverse set of "-ums" ending words ("stadium", "forum", equilibrium", etc).

Let us now consider the list of putative monosemic words (on the right part of Table 5.8), ie. words for which the nearest neighbors are the same in all domains. This list contains mostly words for which we do not expect much variation in translation: adjectives ("previous", "better"), adverbs ("virtually"), generic verbs ("handling", "coordinated"). Further down this list, we will also find prepositions ("at", "in"), auxiliary ("been") etc.

5.6 Conclusions and outlook

In this chapter, we have presented two new techniques for multi-domain machine translation: LDR embeddings, which is a result of adapting the "frustratingly easy" idea of Daumé III (2007) to NMT architectures; CDR, which is an extension of LDR to higher layers of the NMT model.

Our experiments have shown that for both architectures (Transformer and RNN) and two language pairs, LDR improves over the simple DC-Tag system and standard Transformer and that it is robust to noise in domain labels. However, the improvement is still limited in a mild situation where the number of domains is low, and there is less heterogeneity. Our second proposal system CDR outperforms many MDMT systems except FT-Res, which is carefully fine-tuned for each domain in a highly heterogeneous MDMT setting. However, this version of CDR handles only a fixed number of domains.

Furthermore, it is noticeable that these results are obtained without impacting the architecture or training complexity, making our approach an effective baseline for further studies in multi-domain translation. We have also shown that LDR can dynamically handle new domains; and that its domain-specific embeddings often reflect differences of "stand-alone" senses.

In our future work, we aim to develop another version of CDR using learnable domain-node allocations which allows the method to accommodate an arbitrary number of domains.

Chapter 6

Multi-domain residual adapters

6.1 Introduction

In *supervised multi-domain adaptation* discussed in Section 3.3, the fine-tuning method (Luong & Manning, 2015; Freitag & Al-Onaizan, 2016) usually achieve the best performance in single domain adaptation. However, fine-tuned models are usually brittle to out-of-domain examples. Therefore, it is only suitable for the "one domain one model" strategy, which requires much effort for maintenance and training. Several recent studies (Vilar, 2018; Wuebker et al., 2018; Michel & Neubig, 2018; Bapna & Firat, 2019b) have proposed more lightweight schemes to perform domain adaptation while also preserving the value of pre-trained models. Our main inspiration is the latter work, whose proposal relies on small *adapter components* (Bapna & Firat, 2019b) that are plugged in each hidden layer. These adapters are trained only with the in-domain data, keeping the pre-trained model frozen. Bapna & Firat (2019b) used the adapters for MT domain adaptation and multilingual MT. Because these additional adapters are very small compared to the size of the baseline model, their use significantly reduces the cost of training and maintaining fine-tuned models while delivering a performance that remains close to that of full fine-tuning.

In this chapter, we conduct a thorough analysis of the adapter model in the context of a multi-domain machine translation task. We notably explore ways to adjust and/or regularize adapter modules to handle situations where the adaptation data is minimal. We also propose and contrast two new variants of the residual architecture: in the first one (*highway residual adapters*), adaptation still affects each layer of the architecture, but its effect is delayed till the last layer, thus making the architecture more modular and adaptive; our second variant (*gated residual adapters*) exploits this modularity and enables us to explore ways to improve performance in the face of train-test data mismatch. We experiment with two language pairs and report results that illustrate the flexibility and effectiveness of these architectures. Our main conclusions are that residual adapters provide a fast and relatively low-cost method (compared to fine-tuned systems) for supervised multi-domain adaptation; our two variants prove as effective as the original adapter

model and open perspectives to make adapted models more robust to label domain errors.

This chapter draws from the following publications: Pham et al. (2020a).

6.2 Residual adapters

In this section, we describe the basic version of the residual adapter architectures (Houlsby et al., 2019; Bapna & Firat, 2019b), as well as two novel variants of this model.

6.2.1 Basic architecture

6.2.1.1 The computation of adapter layers

Our reference architecture is the Transformer model, which we assume contains a stack of layers both on the encoder and the decoder sides. Each layer contains two subparts, an attention layer, and a dense layer. Details vary from one implementation to another, we simply contend here that each layer $i \in \{1 \dots L\}$ (in the encoder or the decoder) computes a transform of a fixed-length sequence of d-dimensional input vectors h^i into a sequence of output vectors h^{i+1} as follows (**LN** denotes the (sub)layer normalization, **ReLU** is the "rectified linear unit" operator):

$$
\begin{aligned}
h_0^i &= \mathbf{LN}(h^i) \\
h_1^i &= \mathbf{W}_{db}^i h_0^i + a_1^i \\
h_2^i &= \mathbf{ReLU}(h_1^i) \\
h_3^i &= \mathbf{W}_{bd}^i h_2^i + a_2^i \\
\bar{h}^i &= h_3^i + h^i.
\end{aligned}
$$

Overall, the i^{th} adapter is thus parameterized by matrices $\mathbf{W}_{db}^i \in \mathbb{R}^{d \times b}$,$\mathbf{W}_{bd}^i \in \mathbb{R}^{b \times d}$, bias vectors $b_1^i \in \mathbb{R}^b$, $b_2^i \in \mathbb{R}^d$, with b the dimension of the adapter . For the sake of brevity, we will simply denote $h_3^i = \text{ADAP}^{(i)}(h^i)$, and $\theta_{\text{ADAP}^{(i)}}$ the corresponding set of parameters.

The "adapted" hidden vectors $\bar{h}_{1 \leq i \leq L-1}^i$, where L is the number of layers, will then be the input of the $(i+1)^{\text{th}}$ layer; $\bar{h}^L$ is passed to the decoder if it belongs to the encoder side, or is the input of output layer if it belongs to the decoder side. Note that zeroing out all adapters enables us to recover the basic Transformer, with $\bar{h}^i = h^i$ for all i.

In the experiments of Section 6.3, we use $2 \times L = 12$ residual adapters, one for each of the $L = 6$ attention layers of the encoder and similarly for the decoder.[1]

[1] In the decoder, the stack of self-attention and cross encoder-decoder attention only counts as one attention layer and only corresponds to one residual adapter.

6.2.1.2 Design space and variants

This general architecture leaves open many design choices pertaining to the details of the network organization, the training procedure, and the corresponding objective function.

The first question is the number of adapter layers. While in principle, all Transformer layers can be subject to adaptation, it is nonetheless worthwhile to consider simpler adaptation schemes, which would only alter a limited number of layers. Such strategys might be especially relevant when the training data contains very small domains, as in the experiments of Section 6.3, and for which a complete adaptation may not be necessary or/and or prone to overfitting. This, in turn, raises the issue of which layer(s) to adapt, a question that can be approached in the light of recent analyses of Transformers models, which conjecture that the higher layers encode global patterns with a more "semantic" interpretation, while the lower layers encode local patterns akin to morpho-syntactic information Raganato & Tiedemann (2018).

Beside the reduction of the numbers of adapters in a model, we can apply smooth regularization methods to mitigate overfitting. Our first proposition is to use weight decay (Krogh & Hertz, 1991). The method proposes adding a second term, which is the norm L_2 of the model's parameters, to cross-entropy loss as follows

$$\begin{aligned}\bar{L} &= E_{x\sim\mathscr{D}_e^d, y\sim g^d(x)}[-log(P(y|x,\theta_d))] \\ &+ \lambda \sum_{i\in\{1,..,6\}\otimes\{enc,dec\}} \|\theta_{\text{ADAP}_{(d)}^{(i)}}\|_2\end{aligned}$$

in which d is the target domain, variables $\theta_{\text{ADAP}_{(d)}^{(i)}}$ belongs to θ_d and $\theta_d \setminus \{\theta_{\text{ADAP}_{(d)}^{(i)}}\}_{i\in\{1,..,6\}\otimes\{enc,dec\}}$ are frozen during the training. We reuse the same notations as in Equation 5.1. We denote `FT-Res-WD` the MDMT system with adapter tuned by weight decay.

An alternative scheme is *layer regularization*, which penalizes the output of the adapters, corresponding to the following objective:

$$\begin{aligned}\bar{L} &= E_{x\sim\mathscr{D}_e^d, y\sim g^d(x)}\big[-log(P(y|x,\theta_d)) \\ &+ \lambda \sum_{i\in\{1,..,6\}\otimes\{enc,dec\}} \|\text{ADAP}^{(i)}(h_i(x,y))\|_2\big]\end{aligned}$$

We denote `FT-Res-LR` the MDMT system with adapter tuned by layer regularization.

Finally, another independent design choice relates to the training strategy for adapters. A first option is to generalize supervised domain adaptation to multi-domain adaptation and to proceed in two steps: (a) train a generic model with all the available data; (b) train each adapter layer with domain-specific data, keeping the generic model parameters unchanged. Another strategy is to adopt the view of Dredze & Crammer (2008), where the multi-domain setting is considered as an instance of multi-task learning Caruana (1997)

with each domain corresponding to a specific task. This suggests training all the parameters from scratch, as we would do in a multi-task mode. The generic parameters will still depend on all the available data, while each adapter will only be trained with the corresponding in-domain data.

6.2.2 Highway Residual Adapters

In the basic architecture described in Section 6.2.1, the computation performed by lower-level layers will impact all the subsequent layers. This section introduces a variant of residual adapters, which delays the adaptation of each layer to the last layers (of the encoder and the decoder). We demonstrate our proposal in Figure 6.1. While the basic architecture performs adaptation in sequence, we propose here to perform it in parallel. In this version, only the last hidden vector of the encoder (decoder) is thus modified according to:

$$\bar{h}^L = h^L + \sum_{1 \leq i \leq L} ADAP^i(h^i) \tag{6.1}$$

One obvious benefit of this variant is that it allows us to reuse the hidden vectors h^i of all hidden layers when computing an adapted output for several domains during the inference. In this situation, the forward step only needs to compute the hidden vectors h^i once for the inner encoder layers before an adapted sequence of vectors is computed at the topmost layer. Therefore, we can fine-tune the model to multiple domains at once without recomputing h^i. This variant also opens the way to more parameter sharing across adapters, a perspective that we will not explore further in this work. Instead, we use it to develop a second variation of the adapter model, which is presented in the next section.

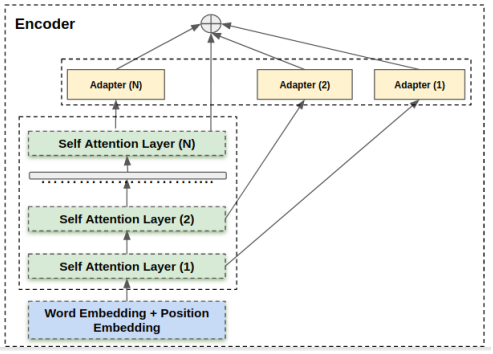


Figure 6.1: Highway residual adapter network

6.2.3 Gated Residual Adapters

The basic architecture presented above rests on a rather simplistic view of "domains" as made of well-separated and unrelated pieces of texts that are processed independently dur-

ing adaptation. Likewise, when translating test documents, one needs to choose between either using one specific domain-adapted model or resorting to the generic model. In this context, using wrong domain labels can have a strong (negative) effect on translation performance.

Therefore, we would like to design a version of residual adapters that is more robust to such domain errors. This variant, called the *gated residual adapter model*, relies on the training of a supplementary component that will help decide whether to activate, on a word per word basis, a given residual layer and to regulate the strength of this activation. To this end, we extend the highway version of residual adapters as follows.

Formally, we replace the adapter computation of equation (6.1) and take the adapted hidden (topmost) layer to be computed as (this is for domain k):

$$\bar{h}^L = h^L + \sum_{1 \leq i \leq L} \text{ADAP}_k^i(h^i) \odot z_k(h^L), \tag{6.2}$$

where the scalar $z_k(h^L[t]) \in [0,1]$ measures the relatedness of the t^{th} word w_t to domain k. The more likely w_t is in domain k, the larger $z_k(h^L[t])$ should be; conversely, for words[2] that are not typical of any domain k (eg. function words), adaptation is minimum and the corresponding adapted encoder output ($\bar{h}^L[t]$) will remain close to the output of the generic model ($h^L[t]$). In our implementation, we incorporate two domain classifiers on top of the encoder and the decoder, that take the last hidden layer of the encoder (resp. decoder) as input and use the posterior probability $P(k|h^L[t])$ of domain k as the value for $z_k(h^L[t])$.

Training gated residual adapters thus comprises three steps, instead of two for the baseline version:

1. learn a generic model with mixed corpora from multiple domains.
2. train a domain classifier on top of the encoder and decoder; during this step, the parameters of the generic model are frozen. This model computes the posterior domain probability $P(k|h^L[t])$ for each word w_t, based on the representation computed by the last layer.
3. train the parameters of adapters with in-domain data separately for each domain, while freezing all the other parameters.

[2]The term "word" is employed here by mere convenience, as systems only manipulate sub-lexical BPE units; furthermore, the values of the hidden representations h^i at position t depend upon all the other positions in the sentence.

6.3 Experimental settings

6.3.1 Data and metrics

We perform our experiments with two translation pairs involving multiple domains: English-French (En→Fr) and English-German (En→De). For the former pair, we reuse the same data as in Chapter 4, which is reported in Section 4.4.1.

En→De is a much larger task, for which we use corpora distributed for the News task of WMT20[3] including: European Central Bank corpus (BANK), European Economic and Social Committee corpus (ECO), European Medicines Agency corpus (MED)[4], Press Release Database of European Commission corpus, News Commentary v15 corpus, Common Crawl corpus (NEWS), Europarl v10 (GOV), Tilde MODEL - czechtourism (TOUR)[5], Paracrawl and Wikipedia Matrix (WEB). Statistics are in Table 6.1.

BANK	ECO	MED	GOV	NEWS	TOUR	WEB
4 (0.00022)	2857 (0.15)	347 (0.018)	1828 (0.095)	3696 (0.19)	7 (0.00039)	10473 (0.54)

Table 6.1: Corpora statistics for En→De: number of parallel lines ($\times 10^3$) and proportion in the basic domain mixture. WEB is the largest domain, containing about 54% of the sentences, while BANK and TOUR are very small.

We randomly select in each corpus a development and a test set of 1,000 lines each and keep the rest for training.[6] Development sets help choose the best model according to the average BLEU score Papineni et al. (2002).[7]

6.3.2 Baseline architectures

Using Transformers implemented in OpenNMT-tf[8] Klein et al. (2017), we train the following baselines:

- a generic model trained on a concatenation of all corpora, denoted `Mixed-Nat`;
- a fine-tuned model (Luong & Manning, 2015; Freitag & Al-Onaizan, 2016), based on the `Mixed-Nat` system, denoted `FT-Full`

The description of these systems and their implementations can be found in Section 4.4.2 and Appendix A.

[3] `http://www.statmt.org/wmt20/news.html`

[4] `https://tilde-model.s3-eu-west-1.amazonaws.com/Tilde_MODEL_Corpus.html`

[5] `https://tilde-model.s3-eu-west-1.amazonaws.com/Tilde_MODEL_Corpus.html`

[6] Scripts to replicate these experiments are available at `https://github.com/qmpham/experiments.git`.

[7] We use detokenized, truecasing and the `multibleu` script.

[8] `https://github.com/OpenNMT/OpenNMT-tf`

For all En→Fr models, we set the embeddings size and the hidden layers size to 512. Transformers use multi-head attention with 8 heads in each of the 6 layers; the inner feedforward layer contains 2,048 cells. Residual adapters additionally use an adaptation block in each layer, composed of a 2-layer perceptron, with an inner ReLU activation function operating on normalized entries of dimension $b = 1024$. Bapna & Firat (2019b) showed that the performance of adapted models increases with respect to the size of the inner dimension and obtained performance close to the full fine-tuned model with $b = 1024$, which is twice as large as the dimension of a Transformer layer. We used the same setting in our experiments.

Training uses a batch size of 12,288 tokens; optimization uses Adam with parameters $\beta_1 = 0.9$, $\beta_2 = 0.98$ and Noam decay ($warmup\_steps = 4,000$), and a dropout rate of 0.1 for all layers. For the `Mixed-Nat` model, we use an initial learning rate of 1.0 and take the concatenation of the validation sets of 6 domains for development. In the fine-tuning experiments, we continue training using `Mixed-Nat` as starting point, using the same learning rate schedule, and continuing the incrementation of the number of steps. In the multi-task training, we use the same learning rate schedule as for `Mixed-Nat`: for each iteration, we sample a domain a probability proportional to its size; we then sample a batch of 12,288 tokens that is used to update the shared parameters and the parameters of the corresponding adapter.

Models for En→De are larger and rely on embeddings as well as hidden layers of size 1024; each Transformers layer contains 16 attention heads; the inner feedforward layer contains 4,096 cells. Adapter modules have the same architecture as for the other language pair, except for their size, which is doubled ($b = 2,048$).

6.3.3 Multi-domain systems

In this section, we evaluate several proposals from the literature on multi-domain adaptation and compare them to full fine-tuning on the one hand, and to two variants of the residual adapter architecture on the other hand. The reference methods included in our experiments are the following:

- a system using "domain control" (Kobus et al., 2017). In this approach, domain information is introduced either as an additional token for each source sentence (`DC-Tag`) or in the form of a supplementary feature for each word (`DC-Feat`);
- a system using lexicalized domain representations presented in Chapter 5 (`LDR`);
- the three proposals of Britz et al. (2017). `TTM` is a feature-based approach where the domain tag is introduced as an extra word *on the target side*. The training uses reference tags and inference is performed with predicted tags, just like for regular target words. `DM` is a multi-task learner where a domain classifier is trained on top of

Model / Domain	MED	LAW	BANK	TALK	IT	REL	AVG
Mixed-Nat	37.3	54.6	50.1	33.5	43.2	77.5	49.4
FT-Full	37.7	59.2	54.5	34.0	46.8	90.8	53.8
DC-Tag	38.1	55.3	49.9	33.2	43.5	80.5	50.1
DC-Feat	37.7	54.9	49.5	32.9	43.6	79.9	49.9
LDR	37.0	54.7	49.9	33.9	43.6	79.9	49.8
TTM	37.3	54.9	49.5	32.9	43.6	79.9	49.7
DM	35.6	49.5	45.6	29.9	37.1	62.4	43.4
ADM	36.4	53.5	48.3	32.0	41.5	73.4	47.5
FT-Res	37.3	57.9	53.9	33.8	46.7	90.2	53.3
MT-Res	37.9	56.0	51.2	33.5	44.4	88.3	51.9
MT-Res$^+$	37.5	57.1	52.4	33.7	46.2	89.5	52.7
MT-Res (gen)	37.7	51.0	34.0	30.4	34.2	15.2	36.4

Table 6.2: Translation performance of various multi-domain MT systems (En→Fr) compared to variants of the residual adapter models. The performance of MDMT contrasts is borrowed from Table 4.3.

the MT encoder, so as to make it aware of domain differences; ADM is the adversarial version of DM, pushing the encoder towards learning domain-independent source representations. These methods only use domain labels in training.

The description and the implementation of each MDMT system are provided in Appendix A and Section 4.4.2.

The two variants of the residual adapter model included in this first round of experiments have been presented in Section 6.2.1: FT-Res is the approach of Bapna & Firat (2019b) based on a two-step training procedure; while MT-Res is the "multi-domain" version, where the parameters of the generic model and of the adapters are jointly learned from scratch. We also report results for the same system, using the parameters of the Mixed-Nat model as initialization (MT-Res$^+$).[9]

Because of the limit of our computational resources, we restrict the experiments in this section to the En→Fr task. Results are in Table 6.2

These results first show that full fine-tuning outperforms all other methods for the in-domain test sets. However, FT-Res is able to reduce the gap with this approach for several domains, showing the effectiveness of residual adapters. The "multi-task" variant is slightly less effective in our experiments than the basic version, where optimization is performed in two steps. As it turns out, using residual adapters proves here better on average than the other reference multi-domain systems; it is also much better than the generic system for translating data from known domains, outperforming the Mixed-Nat system by more than 4 BLEU points in average. Gains are especially large for small domains

[9]This system also includes a layer dropout policy that cancels adapter layers with probability 0.5

such as LAW and REL.

Comparing training schemes (FT-Res vs MT-Res vs MT-Res$^+$) suggests that the simultaneous learning of all parameters is detrimental to performance in our settings: we see that the 2-step procedure implemented in FT-Res always yields the best scores, even when MT-Res is initialized with good parameter values . This may be because in this setting, the adapters have access to a stable version of the generic system. The last line (MT-Res (gen)) gives the results for a MT-Res trained system in which we cancel the adapter in inference - comparing this to Mixed-Nat shows how differently the generic parts of these two systems behave.

6.3.4 Varying the positions and number of residual adapters

Tables 6.3-6.4 report BLEU scores for 6 domains in each language pair: MED, LAW, BANK, TALK, ITand REL for En→Fr; GOV, ECO, TOUR, BANK, MED and NEWS for En→De. We first see that for the latter direction, the basic version FT-Res also outperforms the Mixed-Nat baseline on average, with large gains for the small domains TOUR, BANK and comparable results for the other domains.

By varying the number and position of residual adapters (see Section 6.2.1), we then contrast several implementations. Because the set of possible configurations is large, we only perform experiments for layers $i = 2,4,6$ (both for the encoder and decoder). Two settings are considered: keeping just one adapter or keeping the three. The trend is the same for the two language directions: suppressing adapters always hurts the overall performance, albeit by a small margin: having six adapters is better than three, which is better than keeping only one. However, in the tiniest domain BANK of the $En-De$ pair, which contains only 4k train instances, using less adapters is better than using 6 adapters by a margin of 0.6 BLEU. With only one adapter active, we observe small, insignificant changes in performance when varying the adapter's depth.

6.3.5 Regularizing adapter tuning

The translation from English into German includes two domains (TOUR and BANK) that are extremely small and account only for a very small fraction of the training data (respectively for 0.039% and 0.022% of the total number of sentences). Fine-tuning on these domains can lead to serious overfitting. We assess two well-known regularization techniques for the adapters to help mitigate this problem: weight decay and layer regularization.

For each method, the optimal hyper-parameter λ (weight decay or layer regularization coefficient, see Section 6.2.1.2) are chosen by grid search in a small set of values ($\{10^{-3}, 10^{-4}, 10^{-5}\}$).

Model / Domain	MED	LAW	BANK	TALK	IT	REL	AVG	PARAMS
Mixed-Nat	37.3	54.6	50.1	33.5	43.2	77.5	49.4	65m/0
FT-Res	37.3	57.9	53.9	33.8	46.7	90.2	53.3	65m/12m
FT-Res$_{(2,4,6)}$	37.7	57	53	33.3	45	90	52.7	65m/6m
FT-Res$_{(6)}$	37.7	55.8	51.5	33.9	43.6	89.2	51.9	65m/2m
FT-Res$_{(4)}$	37.9	55.6	51.7	33.7	44.4	88.7	52	65m/2m
FT-Res$_{(2)}$	37.8	55.5	51.4	34	43.8	86.7	51.5	65m/2m
FT-Res-WD	37.2	56.0	52.9	33.4	46.0	90.6	52.7	65m/12m
FT-Res-LR	37.4	56.1	51.8	33.3	45.0	89.7	52.2	65m/12m

Table 6.3: Translation performance of various fine-tuned systems (En→Fr). We report BLEU scores for each domain, as well as averages across domains. Column PARAMS reports the number of domain-agnostic/domain-specific parameters.

Model / Domain	GOV	ECO	TOUR	BANK	MED	NEWS	AVG	PARAMS
Mixed-Nat	29.3	30.5	17.6	38.1	47.9	20.9	30.6	213m/0m
FT-Res	29.6	30.4	19.2	49.0	47.2	20.6	33.1	213m/48m
FT-Res$_{(2,4,6)}$	29.7	30.5	18.8	49.6	47.1	20.6	32.7	213m/24m
FT-Res$_{(6)}$	29.5	30.4	18.1	49.1	46.9	20.4	32.4	213m/8m
FT-Res$_{(4)}$	29.7	30.4	18.1	49.6	47.0	20.6	32.6	213m/8m
FT-Res$_{(2)}$	29.6	30.4	18.3	49.4	46.7	20.6	32.5	213m/8m
FT-Res-WD	29.7	30.8	20.4	50.2	47.7	20.6	33.2	213m/48m
FT-Res-LR	29.6	30.4	19.2	49.0	47.2	20.6	33.1	213m/48m

Table 6.4: Translation performance of various fine-tuned systems (En→De). We report BLEU scores for each domain, as well as averages across domains. Column PARAMS reports the number of domain-agnostic/domain-specific parameters.

Results in Tables 6.3 and 6.4 show that regularizing the adapter model can positively impact the test performance for the smallest domains (this is especially clear for weight-decay (FT-Res-WD) in En→De), at the cost of a small drop in performance for the other domains. Using weight decay proves here to be effective in most cases.

6.3.6 Highway and Gated Residual Adapters

We now turn to the evaluation of our new architectural variants: Highway residual adapters FT-Res-HW on the one hand, and Gated residual adapters FT-Res-Gated on the other hand. We use the same domains and settings as before, focusing here exclusively on the language direction En→Fr.

To also evaluate the robustness with respect to out-of-domain examples, we perform two additional experiments. We first generate translations with erroneous (more precisely: randomly assigned) domain information: the corresponding results appear in Table 6.5 under column RND. We also compute translation for a domain unseen in training (NEWS) as

follows. For each sentence in this test set, we automatically evaluate the closest domain,[10] then use the predicted domain label to compute the translation. This is an error-prone process, which also challenges the robustness of our multi-domain systems. Results are in Table 6.5.

A first observation is that for domains seen in training, our variants FT-Res-HW and FT-Res-Gated achieve BLEU scores that are on a par to those of the original version (FT-Res), with insignificant variations across test sets.

The two other settings are instructive in several ways: they first clearly illustrate the brittleness of domain-adapted systems, for which large drops in performance (more than 15 BLEU points on average) are observed when the domain label is randomly chosen. Our gated variant however proves much more robust than the other adaptation strategy and performs almost on par to the generic system for that test condition. The same trend holds for the unseen NEWS domain, with FT-Res-Gated being the best domain adapted system in our set, outperforming the other variants by about 2 BLEU points.

Model / Domain	MED	LAW	BANK	TALK	IT	REL	AVG	RND	NEWS
Mixed-Nat	37.3	54.6	50.1	33.5	43.2	77.5	49.4	49.4	23.5
FT-Full	37.7	59.2	54.5	34.0	46.8	90.8	53.8	32.5	20.2
FT-Res	37.3	57.9	53.9	33.8	46.7	90.2	53.3	38.4	20.5
FT-Res-HW	37.5	57.2	53.4	33.1	46.3	91.0	53.1	36.6	20.2
MT-Res-HW	37.4	56.4	52.1	33.7	44.8	89.8	52.4	27.1	20.4
MT-Res-HW^{+}	37.7	57.0	52.5	33.5	46.1	89.0	52.6	46.5	21.4
FT-Res-Gate	38.0	57.5	53.0	33.5	46.0	90.1	53.0	49.0	22.5

Table 6.5: Translation performance of highway and gated variants for En→Fr. NEWS is excluded from the training data and considered as an out-of-domain test.

6.4 Conclusions and outlook

In this chapter, we have performed an experimental study of the residual adapter architecture in the context of multi-domain adaptation, where the goal is to build one single system that (a) performs well for domain seen in training, ideally as well as full fine-tuning; (b) is also able to robustly handle translations for new, unseen domains. We have shown that this architecture allowed us to easily adapt a model to a specific domain, delivering BLEU performance than are much better than the generic, mixed domain baseline, thus closing the gap with the full-fine-tuning approach, at a modest computational cost. Several new variants have been introduced and evaluated for two language directions: if none is able to clearly surpass the baseline, residual adapter models, they provide directions for improving this model in practical settings: unbalanced data condition, noise in

[10] As measured by the perplexity of a language model trained with only in-domain data..

label domains, etc. In our future work, we would like to continue the development of the gated variant, which, it seems to us, provides a flexible and robust tool to address the various challenges of multi-domain machine translation.

Chapter 7

Dynamic sampling strategies for multi-domain adaptation

7.1 Introduction

Building effective Neural Machine Translation models often implies accommodating diverse sets of heterogeneous data so as to optimize performance for the domain(s) of interest. Such multi-source / multi-domain adaptation problems are typically approached through instance selection or reweighting strategies, based on a static assessment of the relevance of training instances with respect to the task at hand. In this work, we study dynamic data selection strategies that are able to automatically re-evaluate the usefulness of data samples and to evolve a data selection policy in the course of training. Based on the results of multiple experiments, we show that such methods constitute a generic framework to automatically and effectively cater to a variety of real-world situations, from supervised multi-domain adaptation to unsupervised domain adaptation.

As we usually reported in the previous chapters, MDMT train data usually admit an adverse heterogeneity in the domains' data. In the MDMT setting presented in Section 4.4.1, most common MDMT systems hardly achieve significant improvement compared to the generic model. This might prove the brittleness of the MDMT model-centric approaches to the heterogeneity of data. To overcome this challenge, we study the efficacy of the data-centric approaches, and particularly the data sampling methods.

As introduced in Chapter 3, data sampling approaches aim to find the optimal balance of training data with respect to a certain objective, which can be one or multiple target domains. This is, however, a challenging task due, for instance, to the similarity between domains/languages, but also due to regularization effects of out-of-domain data (Miceli Barone et al., 2017). It may also be suboptimal, as some target domains or languages might be easier to train than others. Finally, improving the performance of the MT system in one domain will often hurt that of another (van der Wees et al., 2017; Britz et al., 2017), and improving model generalization across all domains (Koehn et al., 2018) may

not achieve optimally for any particular domain. Several recent proposals have explored ways to instead consider *dynamic* data selection and sampling strategies that surpass static strategies. Notably, van der Wees et al. (2017); Zhang et al. (2019) construct a static curriculum, while Graves et al. (2017); Platanios et al. (2019); Kumar et al. (2019); Wang et al. (2020b,c) build curricula that automatically adapt to the training data.

In this chapter, we contribute to this line of research in several ways. First, we propose a novel framework (*Multi-Domain Automated Curriculum*, MDAC for short) that simultaneously accounts for the domain adaptation and the multi-domain adaptation problems. Second, we propose a variant of the work of Wang et al. (2020c) for building an automated curriculum for training Multi-Domain NMT models. We evaluate our method in various single-domain and multi-domain adaptation settings. We compare our method with several contrasts, including the work of Zhang et al. (2019) and Wang et al. (2020c), which was previously only applied to multilingual MT. Based on these experimental results, our main conclusions are that (a) using MDAC often yields overall performance that is as good as the standard fine-tuning strategy for domain adaptation; (b) MDAC can effectively handle a variety of test situations, from targeting one single domain to the full multi-domain scenario.

7.2 Learning with multiple data sources

First, we want to recall our definition of MDMT setting in Section 4.2.1. Our train instances are distributed according to a mixture $\mathscr{D}_e^S$ such that $\mathscr{D}_e^S(x) = \sum_{d=1}^{n_d} \lambda^s(d)\mathscr{D}_e^d(x)$, with $\{\lambda^s(d), d = 1 \dots n_d\}$ the mixture weights satisfying $\sum_d \lambda^s(d) = 1$. In the sequel, we also use boldface to denote the vector $\boldsymbol{\lambda}$, and $\lambda(d)$ is used to denote a scalar corresponding to the d^{th} element of the vector $\boldsymbol{\lambda}$.

The main challenge is then to make the best of this heterogeneous data, with the aim to achieve the optimal performance for the intended test conditions. These might correspond to data from just one of the training domains, as in standard supervised (multi-source) domain adaptation; a more difficult case is when the test data is from one domain unseen in training (unsupervised domain adaptation); in multi-domain adaptation finally, the test distribution is itself a mixture of domains, some of which may also be observed in training. Without loss of generality, one may then assume that the test distribution takes the form $\mathscr{D}_e^T(x) = \sum_{d=1}^{n_d} \lambda^t(d)\mathscr{D}_e^d(x)$ - with only one non-null component in the case of pure domain adaptation. These settings are illustrated on Figure 7.1.

These situations have been amply documented from a theoretical perspective (eg. Mansour et al. (2009b,a); Hoffman et al. (2018)). A general recommendation in the DA setting is to adjust the sampling distribution used to optimize the system so as to compensate

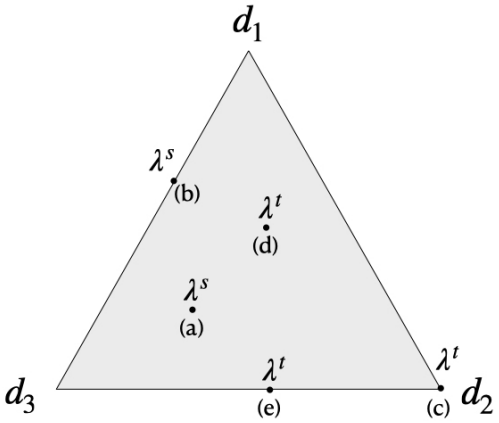


Figure 7.1: Training and testing with distribution mismatch. We consider just three domains, and represent vectors of mixture weights $\boldsymbol{\lambda}^s$ and $\boldsymbol{\lambda}^t$ in the 3-dimensional simplex. Training with weights in (a) and testing with weights in (c) is supervised multi-source domain adaptation to domain 2 (d_2), while (b)-(c) is the unsupervised version, with no training data from d_2; training with weights in (a) and testing with weights in (d) is multi-domain learning, also illustrated with configurations (a)-(e) (training domain d_1 is not seen in test), and (b)-(d) (test domain d_2 is unseen in training).

for the mismatch between $\mathscr{D}_e^S(x)$ and $\mathscr{D}_e^T(x)$. This can be approximated by reweighting instances, or more conveniently domains, which are selected during training with a probability $\lambda^l(d)$, with $\lambda^l(d) \neq \lambda^s(d)$.

A widely used approach to supervised DA is *fine-tuning* (Luong & Manning, 2015; Freitag & Al-Onaizan, 2016), where $\boldsymbol{\lambda}^l$ is allowed to vary during learning. With our notations, this approach amounts to first learning an initial parameter value with all the data ($\forall d, \lambda^l(d) = \lambda^s(d)$), then to continue training with only batches from the test domain d_t ($\lambda^l(d) = \mathbb{I}(d = d_t)$) with $\mathbb{I}(A)$ the indicator function for predicate A. Note that this strategy is potentially suboptimal, as some out-of-domain samples may contribute to the final performance due to eg. domain overlap. Optimizing the learning distribution in multi-domain settings is even more challenging, as the learner has to best take advantage of potential domains overlaps, and also of the fact that some domains might be easier to learn than others.

7.3 Multi-Domain Automated Curriculum

7.3.1 Basic principles

Assuming training data in each of the n_d domains $d_1 \ldots d_{n_d}$, we denote the size of the training corpus from domain d as N_d^s, and $N^s = \sum_d N_d^s$ is the total number of training samples. We use $\widehat{\mathscr{D}_e^{d,l}}$ and $\widehat{\mathscr{D}_e^{d,t}}$ to denote the empirical train and test distributions for

domain d over the source language e, and $\widehat{\mathscr{D}_e^u}(x;\boldsymbol{\lambda}^u) = \sum_d \lambda^u(d)\widehat{\mathscr{D}_e^{d,u}}(x)$ for $u \in \{l,t\}$. In our setting, $\boldsymbol{\lambda}^t$, and hence $\widehat{\mathscr{D}_e^t}(x;\boldsymbol{\lambda}^t)$ are fixed and predefined, approximated with an equivalent number of development corpora.

MDAC constructs an adaptative training distribution $\boldsymbol{\lambda}^l$ that optimizes the data selection policy along with the training of the NMT model. We parameterize $\boldsymbol{\lambda}^l$ by a differentiable function $\boldsymbol{\lambda}^l(\boldsymbol{\psi})$. We divide the training into many short sessions; in each session t, the model is trained with a static data distribution $\boldsymbol{\lambda}^l(\boldsymbol{\psi}_t)$. After one learning session, we update the data distribution using the REINFORCE algorithm of Williams (1992). The evolution of $\boldsymbol{\psi}$ is thus defined by:

$$\boldsymbol{\psi}_{t+1} = \boldsymbol{\psi}_t + \mathbf{lr}_1 * \sum_{d=1}^{n_d} R(d) * \frac{\partial \lambda^l(d;\boldsymbol{\psi}_t)}{\partial \boldsymbol{\psi}},$$

where reward $R(d)$ is computed as:

$$R(d) = J^t(\theta_t,\boldsymbol{\lambda}^t) - J^t(\theta_{t+k},\boldsymbol{\lambda}^t),$$

and where we also define:

$$\begin{aligned}
\theta_{t+i} &= Update\big(\theta_{t+i\text{-}1}, [x_j^i, y_j^i]_{j=1}^N\big) \\
i &\in \{1,\cdots,k\} \\
x_j^i &\sim \widehat{D_e^{d,l}}, y_j^i \sim g^d(x_j^i) \\
J^t(\theta,\boldsymbol{\lambda}_t) &= \sum_{d=1}^{n_d} \lambda^t(d) \mathbb{E}_{x^{d,t}\sim\widehat{D_e^{d,t}}, y^{d,t}\sim g^d(x^{d,t})}\big[l(\theta,x^{d,t},y^{d,t})\big].
\end{aligned}$$

In these equations, k is number of simulation step, N denotes the size of a batch; $\mathbf{lr}_1$ is the learning rate of the sampling distribution; $l(\theta,x,y) = -\sum_i^{l_y} logP(y_i|y_{<i},x;\theta)$ is the loss of the NMT model on sample (x,y); $J^t(\theta,\boldsymbol{\lambda}_t)$ is the weighted loss aggregated over n_d dev-sets corresponding to the n_d domains.

To compute the reward $R(d)$ of using data from domain d to train the model, we simulate k training steps from the current checkpoint by training the model with k batches sampled from $D^l(d)$ and compute the gain of the weighted dev-loss. This computation is inspired by the target prediction gain in Graves et al. (2017). However, where Graves et al. (2017) used accumulated gains from the past as rewards, we instead predict the usefulness of each domain for improving the future performance of the system given its current state. This is achieved by simulating a round of training with only the data from one domain. Furthermore, the choice of the parameterization of the sampling distribution

in our method differs from that of Graves et al. (2017), although this choice is empirical.

The work of Wang et al. (2020c) is also related: it is based on the Bi-level Optimization framework, which aims to find an optimal static distribution $\boldsymbol{\lambda}^{l}$ that will result in the best NMT model with respect to a given target dev set at the end of training. These authors also derive a similar form of update for $\boldsymbol{\psi}$. However, their reward is the cosine similarity between the gradient computed with the training data from one domain and the gradient computed with the dev set. We compare this approach with ours in the experiment section.

7.3.2 MDAC for (multi) domain adaptation

The setting developed in previous sections is quite general and can, in principle, accommodate the variety of situations mentioned above, and many more: basic domain adaption, multi-domain adaptation with various target distributions, possibly including domains unseen in training. In our experiments, we would like to better assess the true potential of MDAC in these settings and seek to experimentally answer the following questions:

- is MDAC a viable alternative to conventional fine-tuning? In particular, does it enable to better take advantage of relevant data from other domains?
- is MDAC also a viable option in multi-domain adaptation scenarios?
- does MDAC also enable to perform *unsupervised* (multi-)domain adaptation?

These questions are further explored in Section 7.5. We now turn to our experimental conditions.

7.4 Experimental settings

7.4.1 Data and metrics

In this work, we reuse the MDMT setting of Section 4.4.1, which has proved to be challenging for MDMT systems. We uses the same metric and processing procedure as in Section 4.4.1. For the sake of brevity, we will not provide the description here.

7.4.2 Baseline systems

Our baselines are standard for multi-domain systems.[1] Using Transformers implemented in OpenNMT-tf[2] (Klein et al., 2017), we build the following systems:

- Generic models trained with predefined mixtures of the training data taking the

[1] We however omit domain-specific systems trained only with the corresponding subset of the data, which are always inferior to the mix-domain strategy (Britz et al., 2017).

[2] `https://github.com/OpenNMT/OpenNMT-tf`

form:

$$\lambda_\alpha(d) = \frac{q_d^\alpha}{\sum_{d=1}^{n_d} q_d^\alpha} \qquad q_d = \frac{|\, N_d^s \,|}{N^s} \tag{7.1}$$

with $\alpha \in \{0, 0.25, 0.5, 0.75, 1.0\}$. We denote these as `Mixed-`$\alpha$ below. `Mixed-0` uses a uniform distribution, `Mixed-1.0` the empirical domain distribution;

- fine-tuned models (Luong & Manning, 2015; Freitag & Al-Onaizan, 2016) which was already presented in Section 4.4.2. Their implementations are provided in Appendix A;
- systems trained with fixed data mixtures corresponding to $\boldsymbol{\lambda}^l \in \left[\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_{0.25}, \boldsymbol{\lambda}_{0.5}, \boldsymbol{\lambda}_{0.75}, \boldsymbol{\lambda}_{1.0}\right]$; these are used in the multi-domain experiments of Section 7.5.3;
- our own implementations of recent dynamic sampling proposals from the literature: Curriculum Learning (CL) of Zhang et al. (2019) and Differential Data Selection (DDS) of Wang et al. (2020c) (see details below);

All models use embeddings and hidden layers of dimension 512. Transformer models contain 8 attention heads in each of the 6+6 layers; the inner feedforward layer contains 2048 cells. Training lasts for 200K iterations, with batches of 12,288 tokens, Adam with parameters $\beta_1 = 0.9$, $\beta_2 = 0.98$, Noam decay ($warmup\_steps = 4000$), and a dropout rate of 0.1 in all layers.

7.4.3 CL and DDS's re-implementation

We re-implement DDS in Tensorflow without any change in the choices of parameterization and hyper-parameters compared to the original code of Wang et al. (2020c).[3] We also re-implement the approach of Zhang et al. (2019) according to the authors' description. For each domain adaptation experiment, we combine the training data of all other domains into one corpus then compute the cross-entropy difference score of each source sentence of this combined dataset. We then sort and split the corpus into 9 shards and execute curriculum learning with 10 shards, using the in-domain corpus as the first shard.

7.4.4 MDAC systems

The behavior of MDAC only depends on (a) the initial domain distribution at the start of training $\boldsymbol{\lambda}^l_{t=0}$, and (b) the targeted (dev/test) distribution $\boldsymbol{\lambda}^t$. We thus report these systems as `MDAC(`$\boldsymbol{\lambda}^l_{t=0}$`,` $\boldsymbol{\lambda}^t$`)` and compare with DDS using the same settings.

In our work, we parameterize the distribution $\boldsymbol{\lambda}^l$ as follows (with $\beta = 2^4$ in all experi-

[3] `https://github.com/cindyxinyiwang/multiDDS`

[4] This setting corresponds to the *spherical softmax* of de Brébisson & Vincent (2016).

ments):

$$\lambda^l(d;\boldsymbol{\psi}) = \frac{\psi[d]^\beta}{\sum_i \psi[i]^\beta}.$$

This parameterization avoids the "rich-get-richer" effect that we observe when using $\boldsymbol{\lambda}(\boldsymbol{\psi}) = \text{softmax}(\boldsymbol{\psi})$, which yields gradients wrt. $\psi[d]$ that are proportional to $\exp(\psi[d])$ (see also Figure 7.2). Additional settings for the hyper-parameters of our method include the number of simulation steps $k = 10$ and the learning rate $\mathbf{lr}_{data} = 0.001$. We update the sampling distribution via 100 gradient descent iterations for almost all experimental settings except that for adaptation with automatic clusters (Section 7.5.5), where we use 20 gradient descent iterations to avoid converging to degenerate distributions. We split the training into 100 short sessions that last 2000 training steps each. The choice of those hyper-parameters is mostly heuristic except for the learning rate $\mathbf{lr}_{data}$ which is optimized via grid search over a set of values $\{0.001, 0.0025, 0.005\}$.

The computational cost of our approach is due to the simulation step, which is conducted after every 2000 iterations to compute the reward of each domain. During the simulation step, we update the temporary checkpoint with k updates for each domain, which cost as much as k training updates. Therefore, we execute $k \times n_d$ updates in total after every 2000 iterations. Our algorithm approximately costs $1 + \frac{k \times n_d}{2000}$ times as much as a standard training.

7.4.5 Experimental tasks

We evaluate and compare our method with multiple baselines in the 5 following conditions. In the *supervised domain adaptation task*, given the data from 6 domains (MED, BANK, LAW, IT, TALK, REL), we aim to build distinct expert NMT models for each domain. To challenge the flexibility of the method, we also consider a *bi-domain adaptation task*, where given the same 6 domains, we only focus on adapating only to 2 domains.

In the *multi-domain adaptation task*, given the same 6 domains, we aim to build one single NMT model that would perform optimally, assuming a uniform distribution of domains during the test.

In a fourth experiment (*unseen domain adaptation*), given training data in 6 domains and a small development set of a new domain (NEWS in our case), we aim to build an NMT model which performs well for the unseen domain.

Finally, in the *unsupervised domain adaptation task*, we cluster all available training data into 30 clusters using the KNN algorithm in the same way as in (Tars & Fishel, 2018), then adapt these clusters to one of 6 domains using the corresponding in-domain dev set. We compare MDAC to DDS for each of our 6 test sets.

7.5 Results and discussion

7.5.1 Domain Adaptation

In this setting, we aim to build an NMT model for one single domain: we accordingly set $\boldsymbol{\lambda}^t$ to a deterministic distribution $\boldsymbol{\lambda}_d$, where the target domain d has probability 1.

We consider three initializations for MDAC and DDS, using $\boldsymbol{\lambda}_0$, $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_d$. According to Table 7.1, MDAC achieves the overall best performance when $\boldsymbol{\lambda}_{t=0} = \boldsymbol{\lambda}_0$. Doing so proves much better than initializing with $\boldsymbol{\lambda}_d$ for small domains: TALK, BANK andIT. Conversely, initializing with $\boldsymbol{\lambda}_d$ is beneficial when targeting large domains such as MED and LAW. The same conclusion holds for DDS.

We now compare the best MDAC system (using $\boldsymbol{\lambda}_{t=0} = \boldsymbol{\lambda}_0$) to full fine-tuning. According to Table 7.1, fine-tuning is better for large domains such as MED and LAW, while MDAC outperforms fine-tuning by approximately 1.2 BLEU for BANK and 1.0 BLEU for REL. This indicates that for small domains, out-of-domain data helps improve the generalization and that MDAC is able to exploit both the in-domain and the out-of-domain training data instead of edging out the out-of-domain training data as in fine-tuning. Results for DDS display similar trends but are always outperformed by MDAC. The behavior of CL, which does only well the large domain MED lag somewhat behind.

domain $d=$	MED	LAW	BANK	TALK	IT	REL	avg.
FT-Full(d)	40.3	63.8	54.4	38.5	52.0	91.0	56.7
CL(d)	40.2	60.2	53.7	36.5	51.1	91.1	55.5
DDS($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_d$)	39.6	60.1	55.0	38.5	52.5	92.0	56.3
MDAC($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_d$)	39.6	62.5**	55.6*	38.5	52.4	92***	56.8
DDS($\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_d$)	39.7	53.9	49.6	37.9	43.1	64.3	48.1
MDAC($\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_d$)	40.2	59.9	52.6	38.5	50.7	79.8	53.6
DDS($\boldsymbol{\lambda}_d, \boldsymbol{\lambda}_d$)	39.9	63.9	54.5	35.4	51.2	91.8	56.1
MDAC($\boldsymbol{\lambda}_d, \boldsymbol{\lambda}_d$)	40.6	63.9	54.5	35.6	51.3	92.3	56.4

Table 7.1: Domain adaptation experiments. We report BLEU scores of each method for 6 target domains and their average: each column corresponds to a distinct system. (*) MDAC is significantly better than CL, fine-tuning and DDS with $p < 0.05$. (**) MDAC is significantly better than CL and DDS with $p < 0.05$. (***) MDAC is significantly better than CL, fine-tuning with $p < 0.05$.

7.5.2 Bi-domain adaptation

In these control experiments, we showcase the flexibility of dynamic sampling and try to adapt to (arbitrary) pairs of target domains with equal weight and contrast our results with those of DDS. Results are in Table 7.2. Here, MDAC significantly outperforms DDS in two settings (MED+IT and LAW+BANK) while being surpassed in TALK+REL.

7.5.3 Multi-domain adaptation

We now turn to a more realistic scenario and consider multi-domain adaptation, which aims to train one single system with optimal performance averaged over 6 domains. This setting targets a uniform test distribution $\boldsymbol{\lambda}^t = \boldsymbol{\lambda}_0$. In this situation, CL (Zhang et al., 2019) does not apply. We therefore only contrast the performance of MDAC, DDS and several fixed training data distribution $\boldsymbol{\lambda}^l \in \left[\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_{0.25}, \boldsymbol{\lambda}_{0.5}, \boldsymbol{\lambda}_{0.75}, \boldsymbol{\lambda}_{1.0}\right]$, where $\boldsymbol{\lambda}_\alpha$ is defined according to equation (7.1).

We again initialize MDAC and DDS with two distribution $\boldsymbol{\lambda}_0$ and $\boldsymbol{\lambda}_1$. According to Table 7.3, MDAC achieves the best performance with initial (uniform) $\boldsymbol{\lambda}_0$. The same conclusion holds for DDS. For this configuration, MDAC outperforms in average static training distributions including $\left[\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_{0.75}, \boldsymbol{\lambda}_{1.0}\right]$ by a significant margin, and performs slightly better than $\left[\boldsymbol{\lambda}_{0.25}, \boldsymbol{\lambda}_{0.5}\right]$. This indicates that MDAC allows us to skip the somewhat heuristic choice of the optimal training mixture.

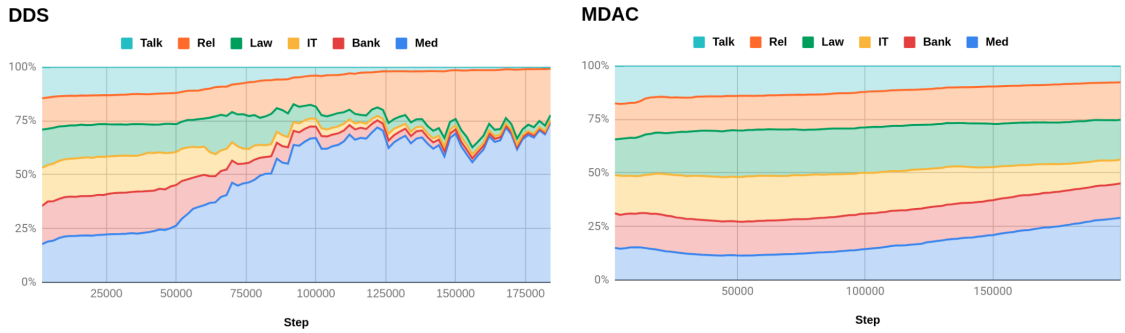
A second observation is that DDS is again outperformed by MDAC by a significant margin of 1.5 BLEU on average; the only domain where DDS does (much) better is MED. Figure 7.2, where we plot the evolution of the training mixture in the course of training sessions, helps understand the difference between the two methods. For DDS (Figure 7.2a), the sampling distribution quickly converges towards a bi-domain mode in which only MED and REL have significant probability – hence the good performance on the former domain. In contrast, the distribution computed by MDAC evolves more smoothly; small domains such as BANK, IT, TALK and REL receive a larger proportion of training data in the early stages; their weights then slowly decrease as larger domains such as MED and LAW increase their share. This, however, only happens at the end of the training, when the NMT models might have already been close to their optimal performance for the small domains.

domain $d =$	MED	LAW	BANK	TALK	IT	REL
DDS($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_2$)	39.5	-	-	-	50.1	-
MDAC($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_2$)	39.1	-	-	-	51.8*	-
DDS($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_2$)	-	60.8	53.3	-	-	-
MDAC($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_2$)	-	61.9*	54.5*	-	-	-
DDS($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_2$)	-	-	-	37.9	-	91.3
MDAC($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_2$)	-	-	-	36.9	-	90.4

Table 7.2: Adapting to two domains. For a given line, non empty columns correspond to the pair of target domains. (*) MDAC is significantly better than DDS with $p < 0.05$.

domain $d=$	MED	LAW	BANK	TALK	IT	REL	mean
Mixed-0	38.6	59.3	53.7	37.3	51.0	90.4	55.1
Mixed-0.25	38.9	59.6	53.3	37.6	50.5	90.6	55.1
Mixed-0.5	39.0	60.2	52.5	38.5	51.9	90.3	55.4
Mixed-0.75	39.4	59.9	51.9	38.8	50.0	87.6	54.6
Mixed-1	40.3	59.5	49.8	36.4	49.0	80.0	52.5
DDS($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_0$)	40.1	56.9	50.7	37.4	46.8	92.0	54.0
MDAC($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_0$)	38.5	60.3**	54.4*	37.3	51.3**	91.4*	55.5**
DDS($\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_0$)	40.6	55.5	48.0	36.2	46.9	60.1	47.9
MDAC($\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_0$)	40.2	59.3**	51.0**	36.9**	48.6**	80.7**	52.8**

Table 7.3: Multi domain adaptation. For a given line, all the columns *correspond to the same multi-domain system*. (*) MDAC is significantly better than Mixed-α with $p < 0.05$. (**) MDAC is significantly better than DDS with $p < 0.05$.



(a) DDS (b) MDAC

Figure 7.2: Evolution of the sampling distribution during training.

7.5.4 Unseen domain

The left part of Table 7.4 displays the performance on the unseen domain NEWS for NMT systems trained with the mixtures $\boldsymbol{\lambda}^l \in [\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_{0.25}, \boldsymbol{\lambda}_{0.5}, \boldsymbol{\lambda}_{0.75}, \boldsymbol{\lambda}_{1.0}]$ and with dynamic data selection (MDAC and DDS). These systems have insignificant differences in

domain $d=$	NEWS
Unseen domain	
Mixed-0	25.7
Mixed-0.25	25.8
Mixed-0.5	26.5
Mixed-0.75	26.8
Mixed-1	26.9
DDS($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_{news}$)	26.3
MDAC($\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_{news}$)	26.3

domain $d=$	MED	LAW	BANK	TALK	IT	REL	mean
Training with 30 clusters							
DDS($\boldsymbol{\lambda}^*, \boldsymbol{\lambda}_d$)	38.3	60.1	50.3	35.8	49.1	90.1	53.9
MDAC($\boldsymbol{\lambda}^*, \boldsymbol{\lambda}_d$)	39.2*	61.6*	52.0*	38.2*	49.1	89.7	55.0*

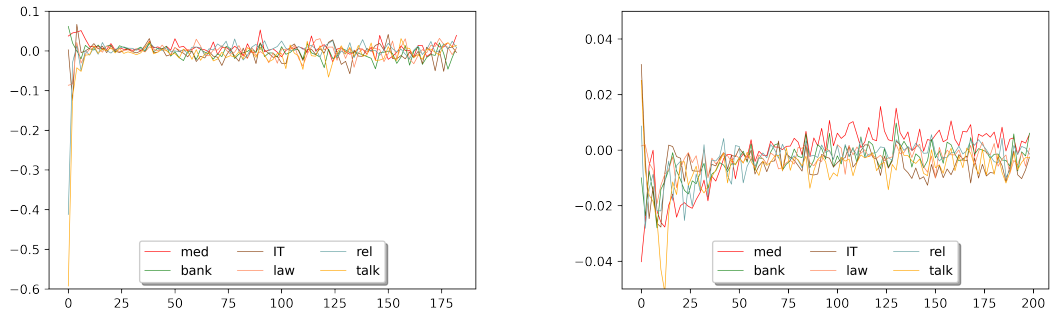
Table 7.4: Unseen domain adaptation (left) and unsupervised adaptation (right). For a given line, each column corresponds to one distinct system. (*) MDAC is significantly better than DDS.

BLEU, showing that dynamic mixture does not improve the robustness of the NMT system against unseen domains. However, the fact that the performance of MDAC and DDS is close to the best performance is a sign that they can also apply in such challenging situations.

7.5.5 Automatic clustering

The right part of Table 7.4 reports the performance of NMT systems adapted to each domain.[5] In comparison to Section 7.5.1, the training data is distributed in 30 automatic clusters instead of the 6 original domains. Splitting the train data into small groups provides the learner with extra degrees of freedom when selecting the best distribution. However, as these clusters are built automatically, they are noisier in nature. According to the results in Table 7.4, this scenario is hard both for DDS and MDAC, which perform much worse than for the supervised DA setting. This again signals the importance of initialization: analyzing the clustering, we find that the data for REL mostly correspond to one single cluster. When using a uniform initialization, this cluster starts with a very small weight and never succeeds in yielding the (good) performance observed in the DA setting.

7.5.6 Reward analysis



(a) DDS (b) MDAC

Figure 7.3: Evolution of the rewards during training.

For MDAC, the magnitude of the rewards decreases dramatically from 1.e-2 to 1.e-3 in about first 10k iterations, then stays around 1.e-3 until the end of the training. We have not tried to rescale the magnitude of rewards to $(-1, 1)$ as proposed in Graves et al. (2017) but left this perspective for the future work.

For DDS, the magnitude of the rewards decreases dramatically from 1.e-1 to 1.e-2 in about first 10k iterations, then stays around 1.e-2 until the end of the training.

[5] A more detailed analysis is in the supplementary material.

7.6 Related Work

Most approaches to adaptive/dynamic data selection take inspiration from Bengio et al. (2009), where the notion of curriculum learning is introduced. CL relies on the notion of the "easiness" of a sample to schedule the presentation of training data so that the easiest examples are presented first and the hardest last. Various ways to automate CL using the framework of multi-armed bandits are explored in (Graves et al., 2017), which has been an inspiration for our implementation. While the initial aim was primarily to improve and speed up training, CL has also proven useful for domain adaptive / multi-domain / multilingual MT, based on alternative definitions of "easiness". For instance, Zhang et al. (2019) study supervised DA and propose a curriculum approach which progressively augments the training data: in the early stages, only in-data is used, while shards containing less relevant[6] data are introduced in later stages. This is somehow opposite to the recommendations of van der Wees et al. (2017), whose *gradual fine-tuning* progressively focuses on the in-domain data.

Kumar et al. (2019) use reinforcement techniques (deep Q-learning) to learn the curriculum strategy: in this work, complexity corresponds to difficulty levels which are binned using contrastive data selection. The reward is based on the decrease of the development set's loss that results from the actual data selection strategy. The same technique is recently applied to multilingual NMT in (Kumar et al., 2021). Zhou et al. (2020) propose another curriculum-based approach which instead relies on *instance uncertainty* as a measure of their difficulty and presents the data sample starting with the easiest (more predictable) first. Another contribution of this work is an alternative criterium for stopping the training. More related to our problems, Wang et al. (2020b) adapt CL for multi-domain adaptation, where an optimal instance weighting scheme is found using Bayesian optimization techniques. Each step consists of (a) weighting instances based on relevance features, (b) fine-tuning a pretrained model using the weighted training set, and is applied iteratively to train a sequence of models. The one that maximizes the development set performance is finally retained.

7.7 Conclusions and outlook

In this study, we have presented a generic framework to perform a variety of standard adaptation tasks for machine translation, ranging from the conventional supervised domain adaptation to multi-domain adaptation and unseen domain adaptation. By experimenting with all these settings, we have shown that the same algorithm, aimed at auto-

[6]Domain distance is computed with Lewis-Moore scores (based on the cross-entropy of in-domain LM and mixed-domain LM).

matically finding an effective data sampling scheme during the course of training, could be used in all these situations. This algorithm, we believe, provides us with a more sound approach to (multi-domain) DA than existing heuristics and dispenses with the costly search of optimal meta-parameters. Another contribution of our work is an experimental comparison of recent approaches to dynamic data selection. In the future, we intend to continue developing this approach and improve its effectiveness. One issue that we have left unaddressed is reward normalization. We remarked that rewards in early stages of the training have much higher magnitude than in the middle and ending stages making premature judgments of the utility of each domain. Kumar et al. (2019) also reported this problem. Graves et al. (2017) proposed a simple rule for re-scaling rewards to the interval $(-1, 1)$. Another area where we need to progress is the unsupervised learning setting of Section 7.5.5, where our results still lag way behind that of supervised DA - this might be due to the inability of our simplistic optimization strategy to handle a large number of domains.

Chapter 8

Attempts at unsupervised multi-domain adaptation

8.1 Introduction

In this chapter, we discuss recent studies on the fourth setting of MDMT presented in Section 3.6. When the test domain is undetermined, we can work around by adapting the generic NMT system to a latent domain of the test sentence on the fly. Li et al. (2018); Farajian et al. (2017b) used similar sentences to adapt the NMT system to the test sentence. The authors proposed finetuning the NMT model over a batch of similar sentences retrieved from a translation memory (TM). We group these method under "one-sentence adaptation paradigm". Recent works of Bulté & Tezcan (2019); Xu et al. (2020) introduced a simple and elegant framework where similar translations are used to improve the context of the translation effectively boosting translation accuracy. We group these methods into the group of "context-augmented NMT". In all cases, context-augmented NMT is performed by simply injecting retrieved sentences in the input stream prior to inference decoding. Context-augmented NMT and one-sentence adaptation can be grouped into a larger category, that we name retrieval-based MT.

In this chapter, our primary goal is to compare two paradigms: on-the-fly one-sentence adaptation (Farajian et al., 2017b; Li et al., 2018) and augmented context NMT (Bulté & Tezcan, 2019; Xu et al., 2020). We also discuss how different retrieval methods affect the quality of context-augmented NMT.

Finally, we propose a variant of Bulté & Tezcan (2019), which performs slightly better and can be used with monolingual data, providing a scenario where NMT can be effectively helped by large amounts of available data. Our proposal does not require to change the NMT architectures or algorithms, relying solely on input preprocessing and on prefix (forced) decoding (Knowles & Koehn, 2016; Santy et al., 2019), a feature already implemented in many NMT toolkits.

8.2 Context-augmented NMT

This section describes the general context-augmented NMT with similar translations. We follow the work by Bulté & Tezcan (2019); Xu et al. (2020) and build a translation model that incorporates similar translations from a translation memory (TM) to boost translation accuracy. In this work, TMs are parallel corpora containing translations falling in the same domain as test sentences.

The following section describes several common retrieval methods used in Farajian et al. (2017b); Bapna & Firat (2019a); Bulté & Tezcan (2019); Xu et al. (2020). We will also describe a variant of Pagliardini et al. (2018), which is developed by Josep Maria Crego.

8.2.1 Similarity Computation

We detail the sentence similarity tools evaluated in this work. The first employs discrete word representations, while the rest relies on building distributed representations of sentences to perform similar sentence retrieval:

`FM`**:** fuzzy matching is a lexicalized matching method aimed to identify non-exact matches of a given sentence. Following Xu et al. (2020), we use `FuzzyMatch`[1], where the fuzzy match score $\text{FM}(s_i, s_j)$ between two sentences s_i and s_j is:

$$\text{FM}(s_i, s_j) = 1 - \frac{\text{ED}(s_i, s_j)}{max(|s_i|, |s_j|)}$$

with $\text{ED}(s_i, s_j)$ being the Edit Distance between s_i and s_j, and $|s|$ is the length of s.

`S2V`**:** we use `sent2vec`[2] (Pagliardini et al., 2018) to generate sentence embeddings. The network implements a simple but efficient unsupervised objective to train distributed representations for sentences. The model is based on efficient matrix factor (bilinear) models (Mikolov et al., 2013a,b; Pennington et al., 2014).

`CBON`**:** the Continuous Bag of n-grams (`CBON`) model denotes our re-implementation of the previous `sent2vec` model. In addition to multiple implementation details, the main difference is the use of arbitrary large n-grams to model sentence representations, where `sent2vec` only used bigrams.

Both `sent2vec` and `CBON` learn a source (or context) embedding $\boldsymbol{v}_w$ for each n-gram

[1] `https://github.com/systran/FuzzyMatch`
[2] `https://github.com/epfml/sent2vec`

w in the vocabulary $\mathscr{V}$. Once the model is trained, the embedding of sentence s (h_s) is obtained as the average of its n-gram embeddings:

$$\boldsymbol{h}_s = \frac{1}{|R(s)|} \sum_{w \in R(s)} \boldsymbol{v}_w$$

where $R(s)$ is the list of n-grams (including unigrams) occuring in sentence s and $\boldsymbol{v}_w$ is the target embedding of n-gram w.

The similarity score $\mathrm{EM}(s_i, s_j)$ between two sentences s_i and s_j is then defined via the cosine similarity of their sentence vector representations h_i and h_j:

$$\mathrm{EM}(s_i, s_j) = \frac{h_i \cdot h_j}{||h_i|| \times ||h_j||},$$

where $||h||$ denotes the norm of vector h.

Note that models differ in their vocabularies, which are built selecting the most frequent n-grams. Both models implement Negative Sampling to avoid the softmax computation.

8.2.2 How to present similar translations to an NMT model

There are several schema to integrate similar translations to an NMT model. We describe below three simple schema, which do not require change in the NMT architecture.

$\texttt{tgt}^k$ this approach was proposed in the work of Bulté & Tezcan (2019), where the input sentence in the source language is augmented with the k translations (in the target language) having the highest matching score (FM or EM) in the TM.

In training, sentence pairs (**s**,**t**) are preprocessed as follows: the source sentence **s** is concatenated with translations t^k of the k most similar sentences (s^k) to **s** found in the TM. Augmented translations are sorted by matching score, with $k = 1$ denoting the most similar. Sentences in the source stream are separated using the special token $\circ$.

$$\begin{aligned} &\texttt{src:} \quad t^k \circ \ldots \circ t^2 \circ t^1 \circ \mathbf{s} \\ &\texttt{tgt:} \quad \mathbf{t} \end{aligned}$$

In inference, only the source-side is input to the translation network.

Xu et al. (2020) reported an issue in the method of Bulté & Tezcan (2019) regarding *unrelated* tokens present in similar translations t^k. The model effectively learns to copy most of the content present in similar translations, but has difficulties to avoid also copying *unrelated* words. Consider for instance the input sentence $s =$ *pertussis vaccin* with similar sentence $s^1 =$ *measles vaccin* and its corresponding translation $t^1 =$ *vaccin contre la rougeole*. Following the $\texttt{tgt}^k$ scheme, the NMT input consists of:

vaccin contre la rougeole ∘ **pertussis vaccin**

yielding the output: **vaccin contre la rougeole**. The word *rougeole* is actually the translation of an unrelated word (*measles*). The model often copies such *unrelated* tokens (Xu et al., 2020), due to the fact that they are present in the input stream as similar translations (t^k) and are usually semantically related to the correct translation choice (here *coqueluche*, the correct translation for *pertussis*).

`tgt`^{k}`+STU` adopts the proposal of Xu et al. (2020) to alleviate the *unrelated word* problem. It relies on an additional source stream (factor) to label related/unrelated tokens. In this scheme the input of the NMT model contains two parallel streams:

$\texttt{src}_1$:	vaccin	contre	la	rougeole	∘	**pertussis**	**vaccin**
$\texttt{src}_2$:	T	T	T	U	T	S	S
tgt:	vaccin	contre	la	coqueluche

Tokens in the second stream are: S for source tokens, U for unrelated and T for related target tokens. *rougeole* is thus tagged as an *unrelated* word that must not be copied in the translation output. Word embeddings are built after concatenating both factor embeddings. Xu et al. (2020) claim achieving a 8% reduction of unrelated tokens when using this scheme.

Note that this solution is computationally expensive as it requires to identify related/unrelated tokens in each input sentence and in the corresponding similar translations, based in Xu et al. (2020) on word alignments and edit distance computations.

`s+t`^{k} the solution introduced in Pham et al. (2020b) relieves us from the tagging burden. It considers both sides of similar translations (s^k and t^k). Training streams take the form:

$$\begin{aligned} \texttt{src:} &\quad s^k \circ \ldots \circ s^2 \circ s^1 \circ \mathbf{s} \\ \texttt{tgt:} &\quad t^k \circ \ldots \circ t^2 \circ t^1 \circ \mathbf{t} \end{aligned}$$

In inference, target-side similar translations t^k are used by the model as a target prefix. The initial steps of the beam search use the given prefix $t^k \circ \ldots \circ t^2 \circ t^1 \circ$ in forced decoding mode, returning to a regular beam search after the last ∘ token is generated.

A similar strategy of concatenating previous and current sentences was explored by Tiedemann & Scherrer (2017) in the context of handling discourse phenomena. However, since we use true translation as prefixes, our strategy does not suffer from exposure bias Ranzato et al. (2016) and the subsequent error propagation problem. Continuing on our running example, during inference the model receives:

`input:` *measles vaccin* ∘ **pertussis vaccin**
`prefix:` *vaccin contre la rougeole* ∘

the encoder embeds the input stream, and force-decodes the target prefix, before starting the translation generation. Note that during beam search, the decoder has thus access both to all input tokens (s^k and s) as well as to similar translations t^k (in the translation prefix).

8.3 Experimental Framework

8.3.1 Corpora

In this chapter, we use an experimental setting larger than the one of Chapter 4. We experiment with the English-French language pair and data originating from eight domains, corresponding to texts from three European institutions: the European Parliament (`EPPS`), the European Medicines Agency (`EMEA`) and the European Central Bank (`ECB`); Legislative texts of the European Union (`JRC`); IT-domain corpora corresponding to `KDE4` and `GNOME`; News Commentaries (`NEWS`); and parallel sentences extracted from Wikipedia (`WIKI`). Table 8.1 contains statistics regarding the corpora used in this work[3] (Tiedemann, 2012). Statistics are computed after splitting off punctuations.

Corpus	#Sents (K)	L_{mean}		Vocab (K)	
		English	French	English	French
Parallel Corpora					
`EPPS`	1,992.8	27.7	32.0	129.5	149.2
`NEWS`	315.3	25.3	31.7	90.5	96.7
`WIKI`	749.0	25.9	23.5	527.5	506.6
`ECB`	174.1	28.6	33.8	45.3	53.5
`EMEA`	336.8	16.8	20.3	62.8	68.9
`JRC`	475.2	30.1	34.5	81.0	83.5
`GNOME`	51.9	9.6	11.6	19.0	21.6
`KDE4`	163.9	9.1	12.4	48.7	64.7
Monolingual Corpora					
`WIKI`	6,426.8	-	24.1	-	1,626.3
`NEWS`	83,567.8	-	25.5	-	3,444.1

Table 8.1: Corpora statistics. Note that K stands for thousands and L_{mean} is the average length in words.

Each corpora is considered as a different domain. Training data sets are also employed as TM of the corresponding domain. This is, similar sentences are mined from the same training set that is used to build the model. Note that we also consider monolingual (French) corpora. For the News domain we use all available monolingual WMT news

[3]Freely available from `http://opus.nlpl.eu`

crawl data[4]. For the Wikipedia domain, we use the French-side of the WikiMatrix data (Schwenk et al., 2021).

We randomly split the parallel corpora by keeping 500 sentences for validation, $1,000$ sentences for testing and the rest for training. All data is preprocessed using the OpenNMT tokenizer[5] (conservative mode).

8.3.2 System Configurations

This section gives learning/inference details of the various systems used in this work.

Similarity

For fuzzy matching `FM` we follow the works of Koehn & Senellart (2010); Bulté & Tezcan (2019); Xu et al. (2020) and keep the n-best matches when $\text{FM}(s_1, s_2) \geq 0.5$ with no approximation. Concerning `S2V`, the model is trained with default options during 20 epochs using all training data. We use an embedding dimension of 300 cells.

Regarding `CBON`, we learn models using also the entire training data during one epoch ($\sim$50,000 iterations). Similarly to `S2V` we use 10 negative samples per positive word to approximate the softmax, a batch size of $2k$ examples, and embedding size of 300 cells. We build `CBON` models using 3-grams and 4-grams to enable a comparison with `sent2vec` which only uses bigrams. All vocabularies are selected keeping the 500,000 most frequent n-grams ($n = 2$ for `S2V` and $n = 3$ and 4 for `CBON`).

For both `CBON` and `S2V` models, we use the 5-best matches when $\text{EM}(s_1, s_2) \geq 0.8$ [6]. In all cases, perfect matches are not used for training. Accuracy results on the translation task indicate that 3-grams yield slightly lower accuracy results than those obtained with 4-grams. In the remainder, we always use the 4-gram version of `CBON`.

Sentence Retrieval

To identify similar translations using distributed representations, we use the `faiss`[7] search toolkit (Johnson et al., 2019) through its Python API with exact *FlatIP* index.

[4] `http://data.statmt.org/news-crawl/`
[5] `https://github.com/OpenNMT/Tokenizer`
[6] Optimization experiments on a held-out development set are carried out for both models.
[7] `https://github.com/facebookresearch/faiss`

Translation

Our NMT models rely on the Transformer base architecture, implemented in the `OpenNMT-tf`[8] toolkit (Klein et al., 2017). We use the standard setting of Transformers for all experiments: size of word embedding: 512; size of hidden layers: 512; size of inner feed-forward layer: 2,048; number of heads: 8; number of layers in the encoder or in the decoder: 6. In the `tgt`^{1}`+STU` scheme, token (508 cells) and `STU` (4 cells) streams are concatenated as in MDMT systems with domain embeddings of Kobus et al. (2017), thus using the same number of parameters in all schemes.

For training, we use the Adam (Kingma & Ba, 2015) optimiser with a batch size of 4,096 tokens. We set the warmup steps to 4,000 and update the learning rate for every 8 iterations. Models are optimised during 300K iterations, using a single NVIDIA V100 GPU. We limit the length of training sentences to 300 BPE tokens (Sennrich et al., 2016b) in both source and target sides to enable the integration of similar sentences. We use a joint BPE-vocabulary of size 32K for both source and target texts. Inference is performed with a beam size of 5 using CTranslate2[9], a custom C++ runtime inference engine for OpenNMT models that enables fast CPU decoding and also implements prefix decoding. For evaluation, we report BLEU (Papineni et al., 2002) scores computed by detokenized, truecasing `multi-bleu.perl`[10].

We re-implement the work of Farajian et al. (2017b) as a contrastive model that we denote μ`adapt`. Note that we only experiment with the basic version of this work, where the closest neighbours of the input sentence are first retrieved from the memory and then used to fine-tune a generic model during 15 additional iterations with a fixed learning rate of 0.0005; the fine-tuned model is then used to produce the translation of the given input sentence. In addition, Farajian et al. (2017b) include a variant where learning rate and number of epochs are dynamically adapted considering sentence similarity. Adaptation is run on a sentence-by-sentence basis.

8.4 Results

Retrieval algorithms employed in this work are significantly faster than NMT Transformer decoding, thus implying a limited decoding overhead.

Table 8.2 reports efficiency scores (tokens/second) for computing vector representations (Vector), performing sentence retrieval (Retrieval) and translation (NMT) for the `WIKI` test set according to the similarity model and context-augmented schema used. Re-

[8] `https://github.com/OpenNMT/OpenNMT-tf`

[9] `https://github.com/OpenNMT/CTranslate2`

[10] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

Model	Schema	Vector	Retrieval	NMT
Base	-	-	-	806
FM	tgt[1]	-	25K	750
	s+t[1]			687
S2V	tgt[5]	222K	17K	639
CBON	tgt[5]	59K		
	s+t[5]			523

Table 8.2: Efficiency (tokens/second) of each step for different inference configurations. All steps run on CPU (16 cores). K stands for thousands.

sults show that the computational cost is dominated by the NMT step. This step, in turn, is affected by the length of the input (and prefix) streams. Table 8.3 reports BLEU scores for various configurations, tested on 8 domain-specific test sets. The last column (avg) reports average results. This table also reports the number of input sentences (out of 1,000) for which at least one similar sentence was retrieved (in a smaller font).

Sim	Scheme	ECB	EMEA	EPPS	GNOME	JRC	KDE4	NEWS	WIKI	avg
Base	-	49.23	49.53	42.83	49.99	59.05	49.52	**36.66**	35.15	46.50
FM	tgt[1]	56.21	59.34	42.08	60.95	65.86	53.49	35.80	34.54	51.03
(Bulté & Tezcan, 2019)		585	765	195	686	612	575	54	184	457
FM	tgt[1]+STU	**57.30**	61.03	42.95	62.68	67.24	54.68	35.54	35.16	52.07
(Xu et al., 2020)		585	765	195	686	612	575	54	184	457
FM	s+t[1]	56.16	60.88	43.18	62.50	67.58	55.25	36.55	36.94	52.38
		585	765	195	686	612	575	54	184	457
S2V	s+t[5]	57.16	60.44	**43.19**	62.44	65.39	51.32	35.98	35.82	51.47
		740	840	161	639	735	623	39	297	509
CBON	s+t[5]	56.50	**61.04**	42.22	**63.76**	**68.75**	**55.83**	35.41	36.38	**52.49**
		710	896	195	854	733	862	63	378	586
FM	μadapt	53.09	55.02	43.04	53.88	62.99	48.70	36.48	35.81	48.63
(Farajian et al., 2017b)		585	765	195	686	612	575	54	184	457
CBON	μadapt	53.41	53.32	43.20	54.77	63.37	52.06	36.47	36.39	49.12
(Farajian et al., 2017b)		710	896	195	854	733	862	63	378	586

Table 8.3: BLEU scores for various model configurations and 8 test domains. Smaller numbers correspond to the number of input sentences in each domain for which at least one similar sentence is found.

All NMT models are built using the concatenation of the original parallel corpora in Table 8.1. The Base configuration does not integrate similar sentences in the training data. All other models extend the original corpora with sentences retrieved following similarity methods (Sim) introduced in Section 8.2.1 and integration schemes presented

in Section 8.2.2 (Scheme).

The second block of results in Table 8.3 displays scores obtained when performing translations extended with fuzzy matches FM. In line with results presented by Xu et al. (2020), using a second stream to mark related/unrelated tokens (+STU) yields a boost in performance of around 1 BLEU points. When the s+t^1 scheme is used, the average improvement reaches 1.25 BLEU points.

The third block compares translation results obtained when identifying similar translations by S2V and CBON. In both cases, the s+t^5 scheme is used. The choice for 5-best similar translations and EM$(s_i, s_j) \geq 0.8$ threshold is made after running optimization work on a held out development set. Sentences identified by CBON outperform those selected by S2V. The idiosyncrasy of fuzzy matching does not enable to find multiple similar sentences for a given input sentence. Overall best results are obtained by the CBON s+t^5 configuration. Note that as expected, the number of similar translations found using distributed representations is larger than those found by fuzzy matching.

Finally, the last block in Table 8.3 gives results for a system that retrieves similar sentences to dynamically adapt the model on a sentence-per-sentence basis (Farajian et al., 2017b; Li et al., 2018). We show micro-adaptation results when similar sentences are found by CBON and FM models (μadapt). In our experiments, micro-adaptation does not yield the gains observed with context-augmented methods. As previously stated, the best performing variants of the adaptation method presented in Farajian et al. (2017b) were not included in our comparison. Variants employ a dynamically adapted learning rate and number of epochs.

Monolingual Corpora

Retrieval results shown in Table 8.3 (small font numbers) indicate a reduced number of similar sentences found for some domains (NEWS, EPPS and WIKI). In the context of scarce similar sentences, the boost in translation quality observed for most domains is subsequently reduced. The case of the NEWS domain is particularly harmful since worst results are always obtained when compared to the Base system.

However, very large monolingual collections of texts exist, far exceeding the amount of available parallel corpora. The latter are more expensive to collect and typically only exist for a limited number of domains and language pairs. With the objective to enhance NMT with monolingual corpora, we now apply the methods presented above to monolingual corpora.

We collect monolingual corpora in the target language (French in this work) and translate each sentence back into English to obtain synthetic parallel data. Similar to back-

translation experiments in Sennrich et al. (2016a), we only use original (human-crafted) target-language data. We expect this to add less noise than incorporating synthetic target-language data into the NMT input. Once translated into English, the various context-augmented approaches identify similar synthetic sentences and injects both the synthetic source and original target in the NMT input stream. Note that cross-lingual sentence embedding models exist (Schwenk & Douze, 2017; Sabet et al., 2019; Conneau & Lample, 2019b) but our preliminary experiments using these tools did not show satisfactory results.

Thus, we exploit large collections of French texts for the News and Wikipedia domains (as detailed in Table 8.1) that we translate into English to enable similarity retrieval. Table 8.4 reports BLEU scores obtained by the best performing network CBON following the s+t[5] scheme.

The supplementary number of similar sentences (468 input sentences have similar translations) collected for the WIKI domain over parallel and monolingual[11] corpora (par+mon) yields an improvement of 2 BLEU points. However, very few (97) similar sentences are identified in NEWS domain[12] over nearly 95 million sentences (par+mon), showing a small gain when compared to using only parallel sentences (par). The network does not succeed to outperform the accuracy of the base system. As outlined by Bulté & Tezcan (2019); Xu et al. (2020) the accuracy of context-augmented networks may slightly drop in performance when no similar translations are integrated. He et al. (2021) mitigated this problem by jointly training the NMT model with and without retrieval.

Sim	Scheme	Data	NEWS	WIKI
Base	-	-	**36.66**	35.15
CBON	s+t[5]	par	35.41 63	36.38 378
CBON	s+t[5]	par+mon	36.05 97	**38.20** 468

Table 8.4: Translation performance for the NEWS and WIKI domain test sets using similar sentences retrieved from parallel data (par) and from both parallel and monolingual (par+mon) data. The first two rows correspond to experiments already shown in Table 8.3.

[11] Test French sentences entirely found in monolingual WIKI corpora are not considered as similar translations.

[12] In all cases we consider similar sentences s_i and s_j when $(\mathrm{EM}(s_i, s_j) \geq 0.8)$

8.5 Discussion

Unrelated Words

As previously outlined in Section 8.2, Xu et al. (2020) raised a problem regarding *unrelated* words. It concerns those words that, even through they appear in similar translations, must not be used to translate input sentences. An example of translation with unrelated word is given in Section 8.2.2 where the input sentence with similar translation:

vaccin contre la rougeole ∘ **pertussis vaccin**

is translated as: **vaccin contre la rougeole**, the right translation being: **vaccin contre la coqueluche**. The error is because word *rougeole* is present in the input stream and is semantically related to *coqueluche*. The problem is particularly hurting when it involves keywords, for example, the proper noun "**pertussis vaccin**" in the above example, which convey essential information regarding the meaning of sentences.

The work by Xu et al. (2020), that we denoted tgt^1+STU, obtains an average reduction of these erroneous words in the translation hypotheses of 8%. We conduct the same experiment to analyse the performance of the new scheme s+t^1 introduced in this work. Table 8.5 reports the total number of unrelated words in 1-best similar sentences obtained by fuzzy matching[13]. As can be seen, the scheme s+t^1 further mitigates the apparition of unrelated words in translations, with a drop of -8.3%.

Scheme	ECB	EMEA	EPPS	GNOME	JRC	KDE4	NEWS	WIKI	avg
tgt^1+STU	3,555	2,320	312	1,285	3,515	940	39	344	1,538
s+t^1	3,199	1,985	306	1,195	3,413	845	31	310	1,410
unrelated	6,310	4,405	4,405	2,473	6,309	2,358	236	1,591	3,510

Table 8.5: Number of unrelated words appearing in test sets according to different augmentation schemes. The last row indicates the total number of unrelated words included in 1-best FM similar sentences.

Similarity with Synthetic Sentences

Results in Table 8.4 show a clear boost in performance (∼2 BLEU points) when making use of synthetic translations from the WIKI monolingual data set. We now want to measure the noise introduced by synthetic translations when compared to human translations. Thus, we consider the input sentences of the WIKI test set for which we found similar sentences

[13] We follow the procedure detailed in Xu et al. (2020) to identify related/unrelated words.

in both the parallel (human translation) and monolingual (synthetic translation) corpus (279 sentences).

Results in Table 8.6 show a clear drop in BLEU scores when using synthetic matches. As expected, machine translation quality degrades the results of similarity search which in turns provides less valuable similar translations.

retrieved sentences	WIKI
par (human)	52.50
mon (synthetic)	49.94

Table 8.6: Results for a reduced test set (279 sentences) using CBON when integrating human and synthetic (back-translated) translations. Here, $par = parallel$, $mon = monolingual$

8.6 Conclusions and outlook

We presented a comparison between two paradigms for undetermined train/test domains. Both paradigms rely on similar translations to boost the quality of the prediction of an NMT model. They admit a considerable latency in the retrieval. Moreover, the one-sentence adaptation paradigm uses additional computational time to compute gradient from a retrieved batch and update model for each test sentence. The context-augmented methods have to process much longer input streams (in encoder and/or decoder) which are K times longer than the sole test sentence. We also demonstrate that only extremely similar retrieved translations boost the quality of context-augmented methods, limiting the application cases of the methods. In fact, domains such as NEWS, which does not include similar translations with high scores in Fuzzy-match, or cosine similarity, can not benefit from this technique. In our experimental setting, which test domains are included in train domains, both paradigms outperform generic models and the finetuning method with large margin. Our proposal variant of the work of Bulté & Tezcan (2019) demonstrates a surprising improvement while using synthesis translation pairs, which is worth to be studied further.

Chapter 9

Conclusions

This thesis aimed to clarify the objectives of MDMT, unify the mathematical notions used in MDMT into one system of notations, and finally propose different approaches for the supervised multi-domain adaptation setting. Besides, we discussed several attempts to handle unknown test domains.

The performance of MDMT systems in individual test sets is usually considered a rule of thumb to accept or reject a novel method. Still, they are often assessed by crudely designed experiments. Moreover, the success of an MDMT system does not rely only on the in-domain performance but also on the robustness with respect to unseen domains, the cross-domain heterogeneity, the intra-domain heterogeneity, the growing number of domains, and erroneous domain tags. Our most important contribution has been to build a multi-criteria evaluation that assesses five previous properties of an MDMT system. Those five "axiomatic" properties correspond to many issues reported in the research of (multi-)domain adaptation, which, however, have never been systematically formalized and assessed with well-designed tests. We built an MDMT testbed in $English - French$ inspired by our proposed multi-criteria evaluation. The data in our testbed consists of seven domains collected from OPUS. Furthermore, this collection is highly unbalanced, which is ideal for assessing MDMT systems. We reimplemented and evaluated a large set of popular MDMT methods that have been frequently used as baselines for comparing with novel MDMT methods. We demonstrated that our MDMT testbed was challenging and revealed many weaknesses of MDMT approaches, thus ideal for assessing the future work in MDMT. Furthermore, this work illustrates the need to use a wide range of domains to evaluate MDMT systems.

Secondly, our three proposal MDMT systems, including `LDR`, `CDR` and `FT-Res-Gated`, constitute different ways to partition parameters/nodes between a subset of domain-agnostic ones and a subset of domain-specific ones. Those systems can close the gap with fine-tuning in a well-conditioned MDMT setting while being robust to erroneous domain tags. However, our proposed methods are still far from fulfilling all of our MDMT multi-criteria evaluations.

The domains' unbalanced train data challenges all the model-centric MDMT approaches. Effectively, the performance of an MDMT model in each domain depends largely on how often train instances of that domain are presented to that model. A heuristic fixed sampling of the train domains usually lags behind more sophisticated dynamical sampling strategies (Wang et al., 2020c). Moreover, fixed sampling strategies can not generalize for non-standard test distributions such as uni-domain adaptation, bi-domain adaptation, or unseen domain adaptation. We studied a group of dynamical sampling strategies which are parametrizable and learned during the training. These methods can adapt the sampling strategy to any predefined test distribution. Moreover, we demonstrated that this type of sampling strategy upper-bounds fixed sampling strategies in general cases while achieving good performance in particular circumstances such as uni-domain adaptation, bi-domain adaptation, or unseen domain adaptation.

Besides, we analyzed several approaches for unknown test domains. These approaches rely on text retrieval, which searches for the most similar translations given a source sentence. There are two propositions to consume the retrieved examples: 1) fine-tuning the current NMT model with these examples; 2) Encoding retrieved examples by an additional encoder or simply concatenating to the source sentence. On the one hand, those methods strongly outperform strong MDMT baselines such as fine-tuning in many domains. On the other hand, those approaches rely on a strong similarity of the retrieved examples and admit a considerable latency due to the retrieval search and the processing of the retrieved examples or the fine-tuning step.

Perspectives

Our work revisited the MDMT literature and illustrated several interesting problems that lack attention from the community, such as the robustness with respect to unknown test domains. For the popular supervised MDMT setting, we described five properties of an effective MDMT system. Each of these fundamental requirements opens many possibilities to improve the quality of an MDMT system. Our future work will be to improve these qualities of an MDMT system.

All model-centric approaches share the same underlying idea, which is to distinguish domain-agnostic parameters and domain-specific parameters. However, the heterogeneity in the proximity of the domains leaves an open question on whether the partition of domain-agnostic and domain-specific parameters in one MDMT model can be optimized automatically rather than predefined heuristically. Furthermore, the sampling strategies should be optimized automatically with respect to the test distribution rather than being chosen heuristically. Our study of dynamical sampling strategies is the first step in that direction. The process of searching for the best hyperparameters or the best partitions for

each pair of MDMT "problem-method" should be carried by machine learning approaches such as Reinforcement Learning.

Next, personalized MT is an extreme case of MDMT (Michel & Neubig, 2018) which constitutes an exciting application in the MT industry. This situation rests on adapting an MT system to a large number of translators' writing styles; thus, MDMT systems are a possible solution. It would, however, requires scaling the methods developed in this thesis to thousands of domains, which remains a non-trivial task.

Besides, we have seen in Chapter 8 that retrieval-based MT demonstrated surprisingly good performance in MDMT. However, this paradigm relies on the retrieval process, which is predefined a priori and unrelated to the translation task. That leaves an open avenue for developing retrieval processes dedicated to the retrieval-based models.

Finally, we hope that future work in multi-domain machine translation will rely on our five axiomatic requirements and pay more attention to the experimental design to assess the proposed method's ability accurately. Furthermore, despite the importance of in-domain performance, the robustness against variable test distributions should be equivalently taken into account. As multilingual machine translation, multi-domain machine translation is a promising paradigm for the MT industry and still needs lots of effort to achieve its long-term objective.

Bibliography

Aharoni, R., & Goldberg, Y. (2020). Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 7747–7763). Online: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/2020.acl-main.692 [Cited on page 57.]

Aharoni, R., Johnson, M., & Firat, O. (2019). Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (pp. 3874–3884). Minneapolis, Minnesota: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/N19-1388 [Cited on page 61.]

Arivazhagan, N., Bapna, A., Firat, O., Lepikhin, D., Johnson, M., Krikun, M., Chen, M. X., Cao, Y., Foster, G., Cherry, C., Macherey, W., Chen, Z., & Wu, Y. (2019). Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv e-prints*, *abs/1907.05019*.
URL http://arxiv.org/abs/1907.05019 [Cited on page 67.]

Assylbekov, Z., Takhanov, R., Myrzakhmetov, B., & Washington, J. N. (2017). Syllable-aware neural language models: A failure to beat character-aware ones. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (pp. 1866–1872). Copenhagen, Denmark: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/D17-1199 [Cited on page 27.]

Ataman, D., Negri, M., Turchi, M., & Federico, M. (2017). Linguistically motivated vocabulary reduction for neural machine translation from turkish to english. *The Prague Bulletin of Mathematical Linguistics*, *108*, 331 – 342. [Cited on page 27.]

Axelrod, A., He, X., & Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, (pp. 355–362). Edinburgh, Scotland, UK.: Association for Computational Linguistics.
URL https://aclanthology.org/D11-1033 [Cited on pages 56 and 58.]

Ba, J., Kiros, J., & Hinton, G. E. (2016). Layer normalization. *ArXiv*, *abs/1607.06450*. [Cited on pages 32 and 38.]

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, ICLR. San Diego, CA.
URL https://arxiv.org/pdf/1409.0473.pdf [Cited on pages 25, 29, 35, 36, and 83.]

Bañón, M., Chen, P., Haddow, B., Heafield, K., Hoang, H., Esplà-Gomis, M., Forcada, M. L., Kamran, A., Kirefu, F., Koehn, P., Ortiz Rojas, S., Pla Sempere, L., Ramírez-Sánchez, G., Sarrías, E., Strelec, M., Thompson, B., Waites, W., Wiggins, D., & Zaragoza, J. (2020). ParaCrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 4555–4567). Online: Association for Computational Linguistics.
URL https://aclanthology.org/2020.acl-main.417 [Cited on pages 47 and 48.]

Bapna, A., & Firat, O. (2019a). Non-parametric adaptation for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), (pp. 1921–1931). Minneapolis, Minnesota: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/N19-1191 [Cited on pages 61 and 122.]

Bapna, A., & Firat, O. (2019b). Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (pp. 1538–1548). Hong Kong, China: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/D19-1165 [Cited on pages 24, 50, 53, 61, 72, 96, 97, 102, and 103.]

Baum, E. B., & Wilczek, F. (1987). Supervised learning of probability distributions by neural networks. In *Proceedings of the 1987 International Conference on Neural Information Processing Systems*, NIPS'87, (p. 52–61). Cambridge, MA, USA: MIT Press. [Cited on page 39.]

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010a). A theory of learning from different domains. *Mach. Learn.*, *79*(1–2), 151–175.
URL https://doi.org/10.1007/s10994-009-5152-4 [Cited on page 44.]

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010b). A theory of learning from different domains. *Mach. Learn.*, *79*(1–2), 151–175.
URL https://doi.org/10.1007/s10994-009-5152-4 [Cited on page 70.]

Ben-David, S., Blitzer, J., Crammer, K., & Pereira, F. (2006). Analysis of representations for domain adaptation. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, (p. 137–144). Cambridge, MA, USA: MIT Press. [Cited on page 51.]

Ben-Tal, A., den Hertog, D., De Waegenaere, A., Melenberg, B., & Rennen, G. (2013). Robust solutions of optimization problems affected by uncertain probabilities. *Manage. Sci.*, *59*(2), 341–357.
URL https://doi.org/10.1287/mnsc.1120.1641 [Cited on page 60.]

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, *3*(null), 1137–1155. [Cited on page 29.]

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, (p. 41–48). New York, NY, USA.
URL https://doi.org/10.1145/1553374.1553380 [Cited on page 119.]

Blitzer, J., McDonald, R., & Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, (p. 120–128). USA: Association for Computational Linguistics. [Cited on page 44.]

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. *Proc. of COMPSTAT*. [Cited on page 40.]

Britz, D., Le, Q., & Pryzant, R. (2017). Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, (pp. 118–126). Association for Computational Linguistics.
URL http://aclweb.org/anthology/W17-4712 [Cited on pages 44, 49, 50, 71, 72, 102, 108, and 112.]

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan,

A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.) *Advances in Neural Information Processing Systems*, vol. 33, (pp. 1877–1901). Curran Associates, Inc.
URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf [Cited on page 37.]

Bulté, B., & Tezcan, A. (2019). Neural fuzzy repair: Integrating fuzzy matches into neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (pp. 1800–1809). Florence, Italy: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/P19-1175 [Cited on pages 61, 121, 122, 123, 126, 128, 130, and 132.]

Burlot, F., & Yvon, F. (2018). Using monolingual data in neural machine translation: a systematic study. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, (pp. 144–155). Brussels, Belgium: Association for Computational Linguistics.
URL https://aclanthology.org/W18-6315 [Cited on page 58.]

Carpuat, M., Daumé, H., Fraser, A., Quirk, C., Braune, F., Clifton, A., Irvine, A., Jagarlamudi, J., Morgan, J., Razmara, M., Tamchyna, A., Henry, K., & Rudinger, R. (2013). Domain adaptation in machine translation : Final report.
URL https://www.cis.uni-muenchen.de/~fraser/pubs/DAMT_2012_final_report.pdf [Cited on pages 44 and 93.]

Carpuat, M., Goutte, C., & Foster, G. (2014). Linear mixture models for robust machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, (pp. 499–509). Baltimore, Maryland, USA: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/W14-3363 [Cited on pages 60 and 67.]

Caruana, R. (1997). Multitask learning. *Mach. Learn.*, *28*(1), 41–75.
URL https://doi.org/10.1023/A:1007379606734 [Cited on pages 73 and 98.]

Cauchy, A. (1847). Méthode générale pour la résolution des systèmes d'équations simultanées. *C.R. Acad. Sci. Paris*, *25*, 536–538.
URL https://ci.nii.ac.jp/naid/10026863174/en/ [Cited on page 40.]

Cettolo, M., Girardi, C., & Federico, M. (2012). Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, (pp. 261–268). Trento, Italy. [Cited on page 70.]

Chen, B., Cherry, C., Foster, G., & Larkin, S. (2017). Cost weighting for neural machine translation domain adaptation. In *Proceedings of the First Workshop on Neural Machine Translation*, (pp. 40–46). Association for Computational Linguistics.
URL http://aclweb.org/anthology/W17-3205 [Cited on page 55.]

Cho, K., van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, (pp. 103–111). Doha, Qatar: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/W14-4012 [Cited on pages 25, 26, 29, and 33.]

Chu, C., Dabre, R., & Kurohashi, S. (2017). An empirical comparison of domain adapta-

tion methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (pp. 385–391). Association for Computational Linguistics.
URL http://aclweb.org/anthology/P17-2061 [Cited on page 54.]

Chu, C., & Wang, R. (2018a). A survey of domain adaptation for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING 2018, (pp. 1304–1319). Santa Fe, New Mexico, USA.
URL http://aclweb.org/anthology/C18-1111 [Cited on pages 43 and 48.]

Chu, C., & Wang, R. (2018b). A survey of domain adaptation for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, (pp. 1304–1319). Santa Fe, New Mexico, USA: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/C18-1111 [Cited on pages 43 and 48.]

Chung, J., Cho, K., & Bengio, Y. (2016). A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (pp. 1693–1703). Berlin, Germany: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/P16-1160 [Cited on page 28.]

Clark, J., Lavie, A., & Dyer, C. (2012). One system, many domains: Open-domain statistical machine translation via feature augmentation. In *Proceedings of the 10th Conference of the Association for Machine Translation in the Americas: Research Papers*. San Diego, California, USA: Association for Machine Translation in the Americas.
URL https://aclanthology.org/2012.amta-papers.4 [Cited on page 67.]

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, (p. 160–167). New York, NY, USA: Association for Computing Machinery.
URL https://doi.org/10.1145/1390156.1390177 [Cited on page 29.]

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, *12*(null), 2493–2537. [Cited on page 29.]

Conneau, A., & Lample, G. (2019a). Cross-lingual language model pretraining. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.) *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc.
URL https://proceedings.neurips.cc/paper/2019/file/c04c19c2c2474dbf5f7ac4372c5b9af1-Paper.pdf [Cited on page 37.]

Conneau, A., & Lample, G. (2019b). Cross-lingual language model pretraining. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.) *Advances in Neural Information Processing Systems 32*, (pp. 7059–7069). Curran Associates, Inc.
URL http://papers.nips.cc/paper/8928-cross-lingual-language-model-pretraining.pdf [Cited on page 130.]

Costa-jussà, M. R., Escolano, C., & Fonollosa, J. A. R. (2017). Byte-based neural machine translation. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, (pp. 154–158). Copenhagen, Denmark: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/W17-4123 [Cited on page 28.]

Costa-jussà, M. R., & Fonollosa, J. A. R. (2016). Character-based neural machine transla-

tion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (pp. 357–361). Berlin, Germany: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/P16-2058 [Cited on page 28.]

Currey, A., Mathur, P., & Dinu, G. (2020). Distilling multiple domains for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 4500–4511). Online: Association for Computational Linguistics.
URL https://aclanthology.org/2020.emnlp-main.364 [Cited on page 58.]

Currey, A., Miceli Barone, A. V., & Heafield, K. (2017). Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, (pp. 148–156). Copenhagen, Denmark: Association for Computational Linguistics.
URL https://aclanthology.org/W17-4715 [Cited on page 58.]

Dakwale, P., & Monz, C. (2017). Fine-tuning for neural machine translation with limited degradation across in- and out-of-domain data. In *Proceedings of the 16th Machine Translation Summit (MT-Summit 2017)*, (pp. 156–169). [Cited on pages 53 and 54.]

Daumé III, H. (2007). Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, (pp. 256–263). Prague, Czech Republic: Association for Computational Linguistics.
URL https://aclanthology.org/P07-1033 [Cited on pages 80, 81, and 95.]

de Brébisson, A., & Vincent, P. (2016). An exploration of softmax alternatives belonging to the spherical loss family. In Y. Bengio, & Y. LeCun (Eds.) *Proceedings of the 4th International Conference on Learning Representation*, ICLR. San Juan, Puerto Rico.
URL http://arxiv.org/abs/1511.05042 [Cited on page 113.]

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*. [Cited on page 37.]

Domhan, T., & Hieber, F. (2017). Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (pp. 1500–1505). Copenhagen, Denmark: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/D17-1158 [Cited on page 55.]

Dou, Z.-Y., Hu, J., Anastasopoulos, A., & Neubig, G. (2019). Unsupervised domain adaptation for neural machine translation with domain-aware feature embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (pp. 1417–1422). Hong Kong, China: Association for Computational Linguistics.
URL https://aclanthology.org/D19-1147 [Cited on pages 53 and 56.]

Dredze, M., & Crammer, K. (2008). Online methods for multi-domain learning and adaptation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, (p. 689–697). USA: Association for Computational Linguistics. [Cited on pages 65 and 98.]

Duh, K., Neubig, G., Sudoh, K., & Tsukada, H. (2013). Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (pp. 678–683). Association for Computational Linguistics.

URL http://aclweb.org/anthology/P13-2119 [Cited on page 56.]

Farajian, M. A., Turchi, M., Negri, M., Bertoldi, N., & Federico, M. (2017a). Neural vs. phrase-based machine translation in a multi-domain scenario. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, (pp. 280–284). Valencia, Spain: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/E17-2045 [Cited on pages 66 and 67.]

Farajian, M. A., Turchi, M., Negri, M., & Federico, M. (2017b). Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*, (pp. 127–137). Copenhagen, Denmark.
URL https://www.aclweb.org/anthology/W17-4713 [Cited on pages 61, 64, 66, 121, 122, 127, 128, and 129.]

Finkel, J. R., & Manning, C. D. (2009). Hierarchical Bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (pp. 602–610). Boulder, Colorado: Association for Computational Linguistics.
URL https://aclanthology.org/N09-1068 [Cited on page 66.]

Freitag, M., & Al-Onaizan, Y. (2016). Fast domain adaptation for neural machine translation. *CoRR*, *abs/1612.06897*. [Cited on pages 53, 55, 60, 67, 68, 71, 96, 101, 110, and 113.]

Gage, P. (1994). A new algorithm for data compression. *C Users J.*, *12*(2), 23–38. [Cited on page 26.]

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In D. Precup, & Y. W. Teh (Eds.) *Proceedings of the 34th International Conference on Machine Learning*, vol. 70 of *Proceedings of Machine Learning Research*, (pp. 1243–1252). International Convention Centre, Sydney, Australia.
URL http://proceedings.mlr.press/v70/gehring17a.html [Cited on pages 35, 36, 37, and 39.]

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh, & M. Titterington (Eds.) *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9 of *Proceedings of Machine Learning Research*, (pp. 249–256). Chia Laguna Resort, Sardinia, Italy: PMLR.
URL https://proceedings.mlr.press/v9/glorot10a.html [Cited on page 40.]

Gong, H., Li, X., & Genzel, D. (2021a). Adaptive sparse transformer for multilingual translation. *ArXiv*, *abs/2104.07358*. [Cited on pages 52 and 81.]

Gong, H., Tang, Y., Pino, J., & Li, X. (2021b). Pay better attention to attention: Head selection in multilingual and multi-domain sequence modeling. *ArXiv*, *abs/2106.10840*. [Cited on pages 52, 61, and 81.]

Graves, A., Bellemare, M. G., Menick, J., Munos, R., & Kavukcuoglu, K. (2017). Automated curriculum learning for neural networks. In D. Precup, & Y. W. Teh (Eds.) *Proceedings of the 34th International Conference on Machine Learning*, vol. 70 of *Proceedings of Machine Learning Research*, (pp. 1311–1320). PMLR.
URL http://proceedings.mlr.press/v70/graves17a.html [Cited on pages 109, 111, 112, 118, 119, and 120.]

Gu, J., Bradbury, J., Xiong, C., Li, V. O. K., & Socher, R. (2017). Non-autoregressive neural machine translation. *CoRR*, *abs/1711.02281*.

URL http://arxiv.org/abs/1711.02281 [Cited on page 41.]

Gülçehre, Ç., Firat, O., Xu, K., Cho, K., Barrault, L., Lin, H.-C., Bougares, F., Schwenk, H., & Bengio, Y. (2015). On using monolingual corpora in neural machine translation. *arXiv e-prints*, *abs/1503.03535*.
URL https://arxiv.org/abs/1503.03535 [Cited on page 55.]

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 770–778). [Cited on page 31.]

He, Q., Huang, G., Cui, Q., Li, L., & Liu, L. (2021). Fast and accurate neural machine translation with translation memory. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (pp. 3170–3180). Online: Association for Computational Linguistics.
URL https://aclanthology.org/2021.acl-long.246 [Cited on page 130.]

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.
URL http://arxiv.org/abs/1503.02531 [Cited on page 55.]

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780.
URL https://doi.org/10.1162/neco.1997.9.8.1735 [Cited on page 33.]

Hoffman, J., Mohri, M., & Zhang, N. (2018). Algorithms and theory for multiple-source adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.) *Advances in Neural Information Processing Systems 31*, (pp. 8246–8256). Curran Associates, Inc.
URL http://papers.nips.cc/paper/8046-algorithms-and-theory-for-multiple-source-adaptat
pdf [Cited on pages 66 and 109.]

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. vol. 97 of *Proceedings of Machine Learning Research*, (pp. 2790–2799). Long Beach, California, USA: PMLR.
URL http://proceedings.mlr.press/v97/houlsby19a.html [Cited on page 97.]

Huck, M., Birch, A., & Haddow, B. (2015). Mixed domain vs. multi-domain statistical machine translation. In *Proceedings of Machine Translation Summit XV: Papers*. Miami, USA.
URL https://aclanthology.org/2015.mtsummit-papers.19 [Cited on page 67.]

Huck, M., Riess, S., & Fraser, A. (2017). Target-side word segmentation strategies for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, (pp. 56–67). Copenhagen, Denmark: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/W17-4706 [Cited on page 27.]

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, (p. 448–456). JMLR.org. [Cited on page 32.]

Irvine, A., Morgan, J., Carpuat, M., Daumé, H., & Munteanu, D. (2013). Measuring machine translation errors in new domains. *Transactions of the Association for Computational Linguistics*, *1*, 429–440.
URL https://doi.org/10.1162/tacl_a_00239 [Cited on page 68.]

Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the echo state network approach. *GMD-Forschungszentrum Informationstechnik, 2002.*, *5*. [Cited on page 40.]

Jang, E., Gu, S., & Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.
URL https://openreview.net/forum?id=rkE3y85ee [Cited on page 52.]

Jean, S., Cho, K., Memisevic, R., & Bengio, Y. (2015). On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (pp. 1–10). Beijing, China: Association for Computational Linguistics.
URL https://aclanthology.org/P15-1001 [Cited on page 26.]

Jiang, H., Liang, C., Wang, C., & Zhao, T. (2020). Multi-domain neural machine translation with word-level adaptive layer-wise domain mixing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 1823–1834). Online: Association for Computational Linguistics.
URL https://aclanthology.org/2020.acl-main.165 [Cited on pages 51 and 66.]

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*.
URL https://arxiv.org/pdf/1702.08734.pdf [Cited on page 126.]

Johnson, M., Schuster, M., Le, Q., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F. a., Wattenberg, M., Corrado, G., Hughes, M., & Dean, J. (2017a). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, *5*, 339–351.
URL https://transacl.org/ojs/index.php/tacl/article/view/1081 [Cited on page 61.]

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., & Dean, J. (2017b). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, *5*, 339–351.
URL https://aclanthology.org/Q17-1024 [Cited on page 66.]

Joshi, M., Dredze, M., Cohen, W. W., & Rosé, C. (2012). Multi-domain learning: When do domains matter? In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, (pp. 1302–1312). Jeju Island, Korea: Association for Computational Linguistics.
URL https://aclanthology.org/D12-1119 [Cited on pages 66 and 67.]

Kiefer, J., & Wolfowitz, J. (1952). Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, *23*(3), 462 – 466.
URL https://doi.org/10.1214/aoms/1177729392 [Cited on page 40.]

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio, & Y. LeCun (Eds.) *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
URL http://arxiv.org/abs/1412.6980 [Cited on page 127.]

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In Y. Bengio, & Y. LeCun (Eds.) *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.
URL http://arxiv.org/abs/1312.6114 [Cited on page 52.]

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2016). Overcoming catastrophic forgetting in neural networks. *arXiv e-prints*, (p. arXiv:1612.00796). [Cited on page 54.]

Klein, G., Kim, Y., Deng, Y., Senellart, J., & Rush, A. (2017). Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, (pp. 67–72). Association for Computational Linguistics.
URL http://aclweb.org/anthology/P17-4012 [Cited on pages 71, 86, 101, 112, and 127.]

Knowles, R., & Koehn, P. (2016). Neural interactive translation prediction. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*. [Cited on page 121.]

Kobus, C., Crego, J., & Senellart, J. (2017). Domain control for neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, (pp. 372–378). INCOMA Ltd.
URL https://doi.org/10.26615/978-954-452-049-6_049 [Cited on pages 49, 53, 61, 64, 72, 85, 102, and 127.]

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, (pp. 388–395). Barcelona, Spain: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/W04-3250 [Cited on pages 70 and 85.]

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X: Papers*, (pp. 79–86). Phuket, Thailand.
URL https://aclanthology.org/2005.mtsummit-papers.11 [Cited on page 84.]

Koehn, P., Khayrallah, H., Heafield, K., & Forcada, M. L. (2018). Findings of the WMT 2018 shared task on parallel corpus filtering. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, (pp. 726–739). Belgium, Brussels: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/W18-6453 [Cited on page 108.]

Koehn, P., & Knowles, R. (2017). Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, (pp. 28–39). Vancouver: Association for Computational Linguistics.
URL https://aclanthology.org/W17-3204 [Cited on pages 43 and 44.]

Koehn, P., & Senellart, J. (2010). Convergence of translation memory and statistical machine translation. In *Proceedings of the Second Joint EM+/CNGL Workshop: Bringing MT to the User: Research on Integrating MT in the Translation Industry*, (pp. 21–32). Denver, Colorado, USA: Association for Machine Translation in the Americas.
URL https://aclanthology.org/2010.jec-1.4 [Cited on page 126.]

Koen, P. (2004). Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas: Technical Papers*, (pp. 115–124). Washington, USA: Springer.
URL https://link.springer.com/chapter/10.1007/978-3-540-30194-3_13 [Cited on page 25.]

Krogh, A., & Hertz, J. A. (1991). A simple weight decay can improve generalization. In J. E. Moody, S. J. Hanson, & R. Lippmann (Eds.) *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*, (pp. 950–957). Morgan Kaufmann.
URL http://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization

[Cited on page 98.]

Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. In I. Gurevych, & Y. Miyao (Eds.) *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, (pp. 66–75). Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/P18-1007/ [Cited on pages 26, 27, and 60.]

Kullback, S., & Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79 – 86.
URL https://doi.org/10.1214/aoms/1177729694 [Cited on page 55.]

Kumar, G., Foster, G., Cherry, C., & Krikun, M. (2019). Reinforcement learning based curriculum optimization for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (pp. 2054–2061). Minneapolis, Minnesota: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/N19-1208 [Cited on pages 109, 119, and 120.]

Kumar, G., Koehn, P., & Khudanpur, S. (2021). Learning policies for multilingual training of neural machine translation systems. *CoRR*, *abs/2103.06964*.
URL https://arxiv.org/abs/2103.06964 [Cited on page 119.]

Lample, G., Conneau, A., Denoyer, L., & Ranzato, M. (2018). Unsupervised machine translation using monolingual corpora only. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.
URL https://openreview.net/forum?id=rkYTTf-AZ [Cited on page 29.]

Le, H. S., Allauzen, A., & Yvon, F. (2012). Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (pp. 39–48). Montréal, Canada: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/N12-1005 [Cited on page 29.]

Lee, J., Cho, K., & Hofmann, T. (2017). Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, *5*, 365–378.
URL https://www.aclweb.org/anthology/Q17-1026 [Cited on page 28.]

Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, *3*, 211–225.
URL https://aclanthology.org/Q15-1016 [Cited on page 29.]

Li, B., Wang, Z., Liu, H., Jiang, Y., Du, Q., Xiao, T., Wang, H., & Zhu, J. (2020). Shallow-to-deep training for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 995–1005). Online: Association for Computational Linguistics.
URL https://aclanthology.org/2020.emnlp-main.72 [Cited on page 33.]

Li, J., & Jurafsky, D. (2016). Mutual information and diverse decoding improve neural machine translation. *ArXiv*, *abs/1601.00372*. [Cited on page 60.]

Li, X., Zhang, J., & Zong, C. (2016). Towards zero unknown word in neural machine translation. In *Proceedings of the Twenty-Fifth International Joint Conference on Arti-*

ficial Intelligence, IJCAI'16, (p. 2852–2858). AAAI Press. [Cited on page 26.]

Li, X., Zhang, J., & Zong, C. (2018). One sentence one model for neural machine translation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA).
URL https://www.aclweb.org/anthology/L18-1146 [Cited on pages 61, 121, and 129.]

Ling, W., Trancoso, I., Dyer, C., & Black, A. W. (2015). Character-based neural machine translation. *CoRR*, *abs/1511.04586*.
URL http://arxiv.org/abs/1511.04586 [Cited on page 28.]

Luong, M.-T., & Manning, C. D. (2015). Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*. Da Nang, Vietnam. [Cited on pages 35, 36, 53, 68, 71, 96, 101, 110, and 113.]

Luong, M.-T., & Manning, C. D. (2016). Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (pp. 1054–1063). Berlin, Germany: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/P16-1100 [Cited on page 28.]

Luong, T., Sutskever, I., Le, Q., Vinyals, O., & Zaremba, W. (2015). Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (pp. 11–19). Beijing, China: Association for Computational Linguistics.
URL https://aclanthology.org/P15-1002 [Cited on page 26.]

Machácek, D., Vidra, J., & Bojar, O. (2018). Morphological and language-agnostic word segmentation for nmt. In *International Conference on Text, Speech, and Dialogue*, (pp. 277–284). [Cited on page 27.]

Mansour, Y., Mohri, M., & Rostamizadeh, A. (2009a). Domain adaptation with multiple sources. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.) *Advances in Neural Information Processing Systems 21*, (pp. 1041–1048). Curran Associates, Inc.
URL http://papers.nips.cc/paper/3550-domain-adaptation-with-multiple-sources.pdf [Cited on pages 66, 67, and 109.]

Mansour, Y., Mohri, M., & Rostamizadeh, A. (2009b). Multiple source adaptation and the rényi divergence. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, (p. 367–374). Arlington, Virginia, USA: AUAI Press. [Cited on pages 66, 67, and 109.]

McCloskey, M., & Cohen, N. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, *24*(C), 109–165. [Cited on page 54.]

Miceli Barone, A. V., Haddow, B., Germann, U., & Sennrich, R. (2017). Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (pp. 1489–1494). Association for Computational Linguistics.
URL http://aclweb.org/anthology/D17-1156 [Cited on pages 54, 75, and 108.]

Michel, P., & Neubig, G. (2018). Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (pp. 312–318). Melbourne, Australia: Association for Computational Linguistics.

URL http://aclweb.org/anthology/P18-2050 [Cited on pages 51, 53, 66, 96, 135, and 166.]

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. In Y. Bengio, & Y. LeCun (Eds.) *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.
URL http://arxiv.org/abs/1301.3781 [Cited on pages 29 and 122.]

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, (p. 3111–3119). Red Hook, NY, USA: Curran Associates Inc. [Cited on pages 29 and 122.]

Miller, T. (2019). Simplified neural unsupervised domain adaptation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (pp. 414–419). Minneapolis, Minnesota: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/N19-1039 [Cited on page 44.]

Moore, R. C., & Lewis, W. (2010). Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, (pp. 220–224). Uppsala, Sweden: Association for Computational Linguistics.
URL https://aclanthology.org/P10-2041 [Cited on page 56.]

Müller, M., Rios, A., & Sennrich, R. (2020). Domain robustness in neural machine translation. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, (pp. 151–164). Virtual: Association for Machine Translation in the Americas.
URL https://aclanthology.org/2020.amta-research.14 [Cited on page 60.]

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (p. 807–814). Madison, WI, USA: Omnipress. [Cited on page 32.]

Neubig, G., Dou, Z.-Y., Hu, J., Michel, P., Pruthi, D., & Wang, X. (2019). compare-mt: A tool for holistic comparison of language generation systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, (pp. 35–41). Minneapolis, Minnesota: Association for Computational Linguistics.
URL https://aclanthology.org/N19-4007 [Cited on pages 70 and 85.]

Ng, N., Yee, K., Baevski, A., Ott, M., Auli, M., & Edunov, S. (2019). Facebook FAIR's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, (pp. 314–319). Florence, Italy: Association for Computational Linguistics.
URL https://aclanthology.org/W19-5333 [Cited on page 47.]

Och, F. J., & Weber, H. (1998). Improving statistical natural language translation with categories and rules. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, (pp. 985–989). Montreal, Quebec, Canada: Association for Computational Linguistics.
URL https://aclanthology.org/P98-2162 [Cited on page 41.]

Oren, Y., Sagawa, S., Hashimoto, T. B., & Liang, P. (2019). Distributionally robust language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural*

Language Processing (EMNLP-IJCNLP), (pp. 4227–4237). Hong Kong, China: Association for Computational Linguistics.
URL https://aclanthology.org/D19-1432 [Cited on pages 60 and 67.]

Ott, M., Edunov, S., Grangier, D., & Auli, M. (2018). Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, (pp. 1–9). Brussels, Belgium: Association for Computational Linguistics.
URL https://aclanthology.org/W18-6301 [Cited on page 31.]

Pagliardini, M., Gupta, P., & Jaggi, M. (2018). Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (pp. 528–540). New Orleans, Louisiana: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/N18-1049 [Cited on page 122.]

Papernot, N., Mcdaniel, P., Wu, X., Jha, S., & Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. *2016 IEEE Symposium on Security and Privacy (SP)*, (pp. 582–597). [Cited on page 60.]

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, (pp. 311–318). Philadelphia, Pennsylvania, USA: Association for Computational Linguistics.
URL https://aclanthology.org/P02-1040 [Cited on pages 42, 68, 70, 84, 101, and 127.]

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, (p. III–1310–III–1318). JMLR.org. [Cited on pages 32, 33, and 40.]

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 1532–1543). Doha, Qatar: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/D14-1162 [Cited on pages 29 and 122.]

Pham, M. Q., Crego, J., & Yvon, F. (2021). Revisiting multi-domain machine translation. *Transactions of the Association for Computational Linguistics*, *9*(0), 17–35.
URL https://transacl.org/index.php/tacl/article/view/2327 [Cited on pages 44, 47, 49, 54, and 65.]

Pham, M. Q., Crego, J.-M., Yvon, F., & Senellart, J. (2019). Generic and Specialized Word Embeddings for Multi-Domain Machine Translation. In *International Workshop on Spoken Language Translation*, Proceedings of the 16th International Workshop on Spoken Language Translation (IWSLT). Hong-Kong, China.
URL https://hal.archives-ouvertes.fr/hal-02343215 [Cited on pages 50, 64, 66, 80, and 81.]

Pham, M. Q., Crego, J. M., Yvon, F., & Senellart, J. (2020a). A study of residual adapters for multi-domain neural machine translation. In *Proceedings of the Fifth Conference on Machine Translation*, (pp. 617–628). Online: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/2020.wmt-1.72 [Cited on pages 50 and 97.]

Pham, M. Q., Xu, J., Crego, J., Yvon, F., & Senellart, J. (2020b). Priming neural machine translation. In *Proceedings of the Fifth Conference on Machine Translation*, (pp. 516–

527). Online: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/2020.wmt-1.63 [Cited on pages 61 and 124.]

Philip, J., Berard, A., Gallé, M., & Besacier, L. (2020). Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 4465–4470). Online: Association for Computational Linguistics.
URL https://aclanthology.org/2020.emnlp-main.361 [Cited on page 56.]

Platanios, E. A., Stretcu, O., Neubig, G., Poczos, B., & Mitchell, T. (2019). Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (pp. 1162–1172). Minneapolis, Minnesota: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/N19-1119 [Cited on page 109.]

Poncelas, A., Wenniger, G. M. D. B., & Way, A. (2018). Feature decay algorithms for neural machine translation. In E. A. for Machine Translation (Ed.) *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, (pp. 239–248).
URL http://rua.ua.es/dspace/handle/10045/76084 [Cited on page 57.]

Post, M. (2018). A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, (pp. 186–191). Brussels, Belgium: Association for Computational Linguistics.
URL https://aclanthology.org/W18-6319 [Cited on page 42.]

Provilkov, I., Emelianenko, D., & Voita, E. (2020). BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 1882–1892). Online: Association for Computational Linguistics.
URL https://aclanthology.org/2020.acl-main.170 [Cited on page 27.]

Raganato, A., & Tiedemann, J. (2018). An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, (pp. 287–297). Brussels, Belgium: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/W18-5431 [Cited on page 98.]

Ramponi, A., & Plank, B. (2020). Neural unsupervised domain adaptation in NLP—A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, (pp. 6838–6855). Barcelona, Spain (Online): International Committee on Computational Linguistics.
URL https://aclanthology.org/2020.coling-main.603 [Cited on page 48.]

Ranzato, M., Chopra, S., Auli, M., & Zaremba, W. (2016). Sequence level training with recurrent neural networks. In Y. Bengio, & Y. LeCun (Eds.) *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
URL http://arxiv.org/abs/1511.06732 [Cited on page 124.]

Rebuffi, S.-A., Bilen, H., & Vedaldi, A. (2017). Learning multiple visual domains with residual adapters. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.) *Advances in Neural Information Processing Systems 30*, (pp. 506–516). Curran Associates, Inc.
URL http://papers.nips.cc/paper/6654-learning-multiple-visual-domains-with-residual-ad
pdf [Cited on page 50.]

Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, *22*(3), 400 – 407.
URL https://doi.org/10.1214/aoms/1177729586 [Cited on page 40.]

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). *Learning Representations by Back-Propagating Errors*, (p. 696–699). Cambridge, MA, USA: MIT Press. [Cited on page 40.]

Sabet, A., Gupta, P., Cordonnier, J.-B., West, R., & Jaggi, M. (2019). Robust cross-lingual embeddings from parallel sentences. [Cited on page 130.]

Sajjad, H., Durrani, N., Dalvi, F., Belinkov, Y., & Vogel, S. (2017). Neural machine translation training in a multi-domain scenario. In *Proceedings of the 14th International Workshop on Spoken Language Translation*, IWSLT 2017. Tokyo, Japan.
URL http://arxiv.org/abs/1708.08712 [Cited on pages 60, 64, and 67.]

Salton, G., & Yang, C. S. (1973). On the specification of term values in automatic indexing. *Journal of Documentation*, *29*, 351–372. [Cited on page 57.]

Santos, J., Consoli, B., & Vieira, R. (2020). Word embedding evaluation in downstream tasks and semantic analogies. In *Proceedings of the 12th Language Resources and Evaluation Conference*, (pp. 4828–4834). Marseille, France: European Language Resources Association.
URL https://aclanthology.org/2020.lrec-1.594 [Cited on page 29.]

Santy, S., Dandapat, S., Choudhury, M., & Bali, K. (2019). INMT: Interactive neural machine translation prediction. 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): System Demonstrations*, (pp. 103–108). Hong Kong, China: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/D19-3018 [Cited on page 121.]

Saunders, D. (2021). *Domain Adaptation for Neural Machine Translation*. Ph.D. thesis, Department of Engineering, University of Cambridge. [Cited on pages 43 and 45.]

Saunders, D., Stahlberg, F., de Gispert, A., & Byrne, B. (2019). Domain adaptive inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (pp. 222–228). Florence, Italy: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/P19-1022 [Cited on pages 22, 54, 60, and 67.]

Schwenk, H. (2012). Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of COLING 2012: Posters*, (pp. 1071–1080). Mumbai, India: The COLING 2012 Organizing Committee.
URL https://www.aclweb.org/anthology/C12-2104 [Cited on page 29.]

Schwenk, H., Chaudhary, V., Sun, S., Gong, H., & Guzmán, F. (2021). WikiMatrix: Mining 135M parallel sentences in 1620 language pairs from Wikipedia. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, (pp. 1351–1361). Online: Association for Computational Linguistics.
URL https://aclanthology.org/2021.eacl-main.115 [Cited on page 126.]

Schwenk, H., & Douze, M. (2017). Learning joint multilingual sentence representations with neural machine translation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, (pp. 157–167). Vancouver, Canada: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/W17-2619 [Cited on page 130.]

Sennrich, R. (2012a). Mixture-modeling with unsupervised clusters for domain adaptation in statistical machine translation. In *Proceedings of the 16th Annual conference of the European Association for Machine Translation*, (pp. 185–192). Trento, Italy: European Association for Machine Translation.
URL https://aclanthology.org/2012.eamt-1.43 [Cited on pages 60 and 67.]

Sennrich, R. (2012b). Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, (pp. 539–549). Avignon, France: Association for Computational Linguistics.
URL https://aclanthology.org/E12-1055 [Cited on pages 60 and 67.]

Sennrich, R. (2013). *Domain adaptation for translation models in statistical machine translation*. Ph.D. thesis, University of Zurich.
URL https://doi.org/10.5167/uzh-88574 [Cited on page 45.]

Sennrich, R., Haddow, B., & Birch, A. (2016a). 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)*, (pp. 86–96). Association for Computational Linguistics.
URL http://aclweb.org/anthology/P16-1009 [Cited on pages 58 and 130.]

Sennrich, R., Haddow, B., & Birch, A. (2016b). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (pp. 1715–1725). Berlin, Germany.
URL https://www.aclweb.org/anthology/P16-1162 [Cited on pages 26, 70, 85, and 127.]

Sennrich, R., Schwenk, H., & Aransa, W. (2013). A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (pp. 832–840). Sofia, Bulgaria: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/P13-1082 [Cited on page 66.]

Servan, C., Crego, J. M., & Senellart, J. (2016). Domain specialization: a post-training domain adaptation for neural machine translation. *CoRR*, *abs/1612.06141*.
URL http://arxiv.org/abs/1612.06141 [Cited on page 53.]

Sharaf, A., Hassan, H., & Daumé III, H. (2020). Meta-learning for few-shot NMT adaptation. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, (pp. 43–53). Online: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/2020.ngt-1.5 [Cited on page 72.]

Silva, C. C., Liu, C.-H., Poncelas, A., & Way, A. (2018). Extracting in-domain training corpora for neural machine translation using data selection methods. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, (pp. 224–231). Belgium, Brussels: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/W18-6323 [Cited on page 57.]

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958.
URL http://jmlr.org/papers/v15/srivastava14a.html [Cited on page 40.]

Steinberger, R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufis, D., & Varga, D. (2006). The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, LREC'06. Genoa, Italy: European Language Resources Association (ELRA).

[Cited on pages 47 and 70.]

Su, J., Zeng, J., Xie, J., Wen, H., Yin, Y., & Liu, Y. (2019). Exploring discriminative word-level domain contexts for multi-domain neural machine translation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (pp. 1–1). [Cited on page 72.]

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 2818–2826). Los Alamitos, CA, USA: IEEE Computer Society.
URL `https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.308` [Cited on page 40.]

Tars, S., & Fishel, M. (2018). Multi-domain neural machine translation. In J. A. Pérez-Ortiz, F. Sánchez-Martínez, M. Esplà-Gomis, M. Popović, C. Rico, A. Martins, J. V. den Bogaert, & M. L. Forcada (Eds.) *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, EAMT, (pp. 259–269). Alicante, Spain: EAMT.
URL `https://arxiv.org/pdf/1805.02282.pdf` [Cited on page 114.]

Thompson, B., Gwinnup, J., Khayrallah, H., Duh, K., & Koehn, P. (2019). 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, (pp. 2062–2068).
URL `https://www.aclweb.org/anthology/N19-1209/` [Cited on pages 53 and 54.]

Tiedemann, J. (2009). News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, & R. Mitkov (Eds.) *Recent Advances in Natural Language Processing*, vol. V, (pp. 237–248). Borovets, Bulgaria: John Benjamins, Amsterdam/Philadelphia. [Cited on pages 47, 70, and 84.]

Tiedemann, J. (2012). Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, (pp. 2214–2218). Istanbul, Turkey: European Language Resources Association (ELRA).
URL `http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf` [Cited on pages 70 and 125.]

Tiedemann, J., & Scherrer, Y. (2017). Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, (pp. 82–92). Copenhagen, Denmark: Association for Computational Linguistics.
URL `https://www.aclweb.org/anthology/W17-4811` [Cited on page 124.]

Tu, Z., Liu, Y., Shang, L., Liu, X., & Li, H. (2017). Neural machine translation with reconstruction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, (p. 3097–3103). AAAI Press. [Cited on page 60.]

van der Wees, M. (2017). What's in a domain?: Towards fine-grained adaptation for machine translation. [Cited on pages 43 and 44.]

van der Wees, M., Bisazza, A., & Monz, C. (2017). Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (pp. 1400–1410). Copenhagen, Denmark: Association for Computational Linguistics.
URL `https://aclanthology.org/D17-1147` [Cited on pages 58, 59, 108, 109, and 119.]

van der Wees, M., Bisazza, A., Weerkamp, W., & Monz, C. (2015). What's in a domain? analyzing genre and topic differences in statistical machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, (pp. 560–566). Beijing, China: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/P15-2092 [Cited on pages 43 and 44.]

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.) *Advances in Neural Information Processing Systems 30*, (pp. 5998–6008). Curran Associates, Inc.
URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf [Cited on pages 29, 31, 32, 35, 37, 39, 51, 83, 86, 155, and 156.]

Vilar, D. (2018). Learning hidden unit contribution for adapting neural machine translation models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, (pp. 500–505). Association for Computational Linguistics.
URL http://aclweb.org/anthology/N18-2080 [Cited on pages 50, 53, and 96.]

Wang, C., Cho, K., & Gu, J. (2020a). Neural machine translation with byte-level subwords. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(05), 9154–9160.
URL https://ojs.aaai.org/index.php/AAAI/article/view/6451 [Cited on page 28.]

Wang, R., Finch, A., Utiyama, M., & Sumita, E. (2017a). Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (pp. 560–566). Vancouver, Canada: Association for Computational Linguistics.
URL http://aclweb.org/anthology/P17-2089 [Cited on page 57.]

Wang, R., Utiyama, M., Liu, L., Chen, K., & Sumita, E. (2017b). Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (pp. 1482–1488). Association for Computational Linguistics.
URL http://aclweb.org/anthology/D17-1155 [Cited on page 55.]

Wang, W., Caswell, I., & Chelba, C. (2019). Dynamically composing domain-data selection with clean-data selection by "co-curricular learning" for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (pp. 1282–1292). Florence, Italy: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/P19-1123 [Cited on page 59.]

Wang, W., Tian, Y., Ngiam, J., Yang, Y., Caswell, I., & Parekh, Z. (2020b). Learning a multi-domain curriculum for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 7711–7723). Online: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/2020.acl-main.689 [Cited on pages 109 and 119.]

Wang, X., Tsvetkov, Y., & Neubig, G. (2020c). Balancing training for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 8526–8537). Online: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/2020.acl-main.754 [Cited on pages 52, 61, 109, 112, 113, and 134.]

Welleck, S., Brantley, K., Iii, H. D., & Cho, K. (2019). Non-monotonic sequential text generation. In K. Chaudhuri, & R. Salakhutdinov (Eds.) *Proceedings of the 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, (pp. 6716–6726). PMLR.
URL http://proceedings.mlr.press/v97/welleck19a.html [Cited on page 41.]

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, *8*(3–4), 229–256.
URL https://doi.org/10.1007/BF00992696 [Cited on pages 52 and 111.]

Wuebker, J., Simianer, P., & DeNero, J. (2018). Compact personalized models for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (pp. 881–886). Brussels, Belgium: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/D18-1104 [Cited on page 96.]

Xu, J., Crego, J., & Senellart, J. (2020). Boosting neural machine translation with similar translations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 1580–1590). Online: Association for Computational Linguistics.
URL https://aclanthology.org/2020.acl-main.144 [Cited on pages 61, 121, 122, 123, 124, 126, 128, 129, 130, and 131.]

Yarowsky, D. (1993). One sense per collocation. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, (p. 266–271). USA: Association for Computational Linguistics.
URL https://doi.org/10.3115/1075671.1075731 [Cited on page 93.]

Zeng, J., Su, J., Wen, H., Liu, Y., Xie, J., Yin, Y., & Zhao, J. (2018). Multi-domain neural machine translation with word-level domain context discrimination. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (pp. 447–457). Brussels, Belgium: Association for Computational Linguistics.
URL http://aclweb.org/anthology/D18-1041 [Cited on pages 51, 53, 64, 66, 72, 85, and 88.]

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020). PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In H. D. III, & A. Singh (Eds.) *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, (pp. 11328–11339). PMLR.
URL http://proceedings.mlr.press/v119/zhang20ae.html [Cited on page 37.]

Zhang, X., Shapiro, P., Kumar, G., McNamee, P., Carpuat, M., & Duh, K. (2019). Curriculum learning for domain adaptation in neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (pp. 1903–1915). Minneapolis, Minnesota: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/N19-1189 [Cited on pages 59, 109, 113, 116, and 119.]

Zhou, L., Zhang, J., & Zong, C. (2019). Synchronous bidirectional neural machine translation. *Transactions of the Association for Computational Linguistics*, *7*, 91–105.
URL https://www.aclweb.org/anthology/Q19-1006 [Cited on page 41.]

Zhou, Y., Yang, B., Wong, D. F., Wan, Y., & Chao, L. S. (2020). Uncertainty-aware curriculum learning for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 6934–6944). Online: Association for Computational Linguistics.
URL https://www.aclweb.org/anthology/2020.acl-main.620 [Cited on page 119.]

Appendix A

A - Description of multi-domain systems (Chapter 4)

We use the following setups for MDMT systems.

- `Mixed-Nat`, `FT-full`, `TTM`, `DC-Tag` use a medium Transformer model of Vaswani et al. (2017) with the following settings: embeddings size and hidden layers size are set to 512. Multi-head attention comprises 8 heads in each of the 6 layers; the inner feedforward layer contains 2048 cells. Training use a batch size of $12,288$ tokens; optimization uses Adam with parameters $\beta_1 = 0.9$, $\beta_2 = 0.98$ and Noam decay ($warmup\_steps = 4000$), and a dropout rate of 0.1 for all layers.
- `FT-Res` and `MT-res` use the same medium Transformer and add residual layers with a bottleneck dimension of size 1024.
- `ADM`, `DM` use medium Tranformer model and a domain classifier composing of 3 dense layers of size 512×2048, 2048×2048 and $2048 \times domain\_num$. The two first layers of the classifier use the ReLU() as activation function, the last layer uses tanh() as activation function.
- `DC-Feat` uses medium Transformer model and domain embeddings of size 4. Given a sentence of domain i in a training batch, the embedding of domain i is concatenated to the embedding of each token in the sentence.
- `LDR` uses medium Transformer model and for each token we introduce a `LDR` feature of size $4 \times domain\_num$. Given a sentence of domain $i \in [1,..,K]$ in the training batch, for each token of the sentence, the `LDR` units of the indexes outside of the range $[4(i-1),..,4i-1]$ are masked to 0, and the masked LDR feature will be concatenated to the embedding of the token of size 508. Details are in Section 5.4.3.2.
- `Mixed-Nat-RNN` uses one bidirectional LSTM layer in the encoder and one LSTM layer in the decoder. The size of hidden layers is 1024, the size of word embeddings is 512.
- `WDCNMT` uses one bidirectional GRU layer in the encoder and one GRU-conditional layer in the decoder. The size of hidden layers is 1024, the size of word embeddings

is 512.

Training For each domain, we create train/dev/test sets by randomly splitting each corpus. We maintain the size of validation sets and of test sets equal to 1,000 lines for every domain. The learning rate is set as in Vaswani et al. (2017). For the fine-tuning procedures used for `FT-full` and `FT-Res`, we continue training using the same learning rate schedule, continuing the incrementation of the number of steps. All other MDMT systems reported in Tables 4.3 and 4.4 use a combined validation set comprising 6,000 lines, obtained by merging the six development sets. For the results in Table B.1 we also append the validation set of `NEWS` to the multi-domain validation set. In any case, training stops if either training reaches the maximum number of iterations (50,000) or the score on the validation set does not increase for three consecutive evaluations. We average five checkpoints to get the final model.

Appendix B

B - Experiments with continual learning (Chapter 4)

Domain Model	MED	LAW	BANK	TALK	IT	REL	NEWS	WAVG	AVG
Mixed-Nat	37.1 +0.2 | –	54.1 +0.5 | —	49.6 +0.5 | –	34.1 -0.6 | –	42.1 +1.1 | –	77.0 +0.5 | –	28.9 -5.4 | –	40.8 +0.3 | –	49.0 +0.4 | –
DC-Tag	37.7 +0.3 | +0.3	54.5 **+0.8** | -0.1	49.9 -0.04 | -0.6	34.8 -1.6 | -1.1	43.9 -0.4 | -1.3	78.8 **+1.7** | -3.5	29.5 -7.7 | -1.4	41.4 +0.2 | -0.1	49.9 +0.1 | -1.1
DC-Feat	37.4 +0.3 | -0.2	54.9 -0.1 | -0.1	50.0 -0.3 | -0.1	34.7 -1.3 | -0.6	43.9 -0.1 | -0.9	79.6 +0.4 | +0.3	28.9 -7.3 | -0.8	41.2 +0.1 | -0.2	50.1 -0.2 | -0.3
LDR	37.0 0.0 | -0.6	54.6 +0.1 | +0.5	49.6 +0.2 | -0.4	34.3 -0.4 | -0.6	43.0 +0.5 | +0.5	77.0 **+2.9** | **+3.8**	28.7 -6.6 | -0.9	40.8 +0.6 | +0.5	49.2 +0.1 | -0.4
TTM	37.3 0.0 | -0.3	54.4 +0.4 | -0.3	49.6 -0.1 | -0.5	33.8 -0.9 | -1.1	42.9 +0.6 | -1.0	78.2 **+1.8** | -4.0	29.1 -5.7 | -1.4	41.0 0.0 | -0.5	49.4 +0.3 | -1.2
DM	36.0 -0.4 | +0.6	51.3 -1.8 | +0.4	46.8 -1.2 | +0.6	31.8 -1.8 | -0.1	39.8 -2.6 | +0.5	65.7 -3.3 | 0.0	27.0 -4.4 | -1.2	38.9 -0.8 | +0.5	45.2 -1.8 | +0.3
ADM	36.6 -0.2 | +0.3	54.2 -0.7 | -0.8	49.1 -0.8 | -0.8	32.9 -0.9 | -0.2	42.1 -0.5 | -0.4	75.7 -2.3 | -5.0	28.7 -5.4 | -1.9	40.2 -0.5 | -0.2	48.4 -0.9 | -1.1
FT-Res	37.0 +0.3 | +0.3	57.6 +0.4 | +0.4	53.8 +0.1 | +0.1	34.5 -0.7 | -0.7	46.1 +0.5 | +0.5	91.1 -0.9 | -0.9	29.6 -9.0 | -0.6	42.2 -0.1 | -0.1	53.3 +0.2 | +0.2
MT-Res	37.7 +0.2 | -0.2	55.6 +0.4 | +0.5	51.1 +0.1 | 0.0	34.4 -0.9 | -0.4	44.5 -0.1 | -0.2	87.5 +0.9 | -0.2	29.1 -8.0 | -0.8	41.9 +0.1 | -0.2	51.8 +0.1 | -0.1

Table B.1: Ability to handle a new domain. We report BLEU scores for a complete training session with 7 domains, as well as differences with (left) training with 6 domains (from Table 4.3); (right) continuous training mode. Averages only take into account six domains (NEWS excluded). Underline denotes a significant loss, bold a significant gain.

Appendix C

C - Experiments with automatic domains (Chapter 4)

This experiment aims to simulate with automatic domains a scenario where the number of "domains" is large and where some "domains" are close and can effectively share information. Full results in Table C.1. Cluster size vary from approximately 8k sentences (cluster 24) up to more than 350k sentences. More than 2/3 of these clusters mostly comprise texts from one single domain, as for cluster 12 which is predominantly MED, the remaining clusters typically mix 2 domains. Fine-tuning with small domains is often outperformed by other MDMT techniques, an issue that a better regularization strategy might mitigate. Domain-control (DC-Feat) is very effective for small domains, but again less so in larger data conditions. Among the MD models, approaches using residual adapters have the best average performance.

Model Cluster	size train / test	Mixed Nat	FT Full	FT Res	MT Res	DC Feat	DC Tag	TTM	ADM	DM	LDR
24 [med]	8.1k / 3	90.4	90.4	90.4	90.4	100.0	65.6	100.0	90.4	100.0	100.0
13 [-]	17.3k / 52	67.6	75.4	74.3	74.3	75.0	54.7	74.7	75.9	65.9	76.9
28 [-]	25.6k / 54	71.6	68.7	68.1	70.2	71.0	42.5	72.0	71.3	65.6	72.6
19 [IT]	27.2k / 88	58.5	63.0	60.9	63.9	63.7	57.2	59.4	61.1	60.5	60.3
0 [-]	27.4k / 72	43.9	33.3	45.4	45.4	49.9	15.4	46.8	49.2	46.6	47.8
22 [-]	27.5k / 103	91.5	93.7	93.4	93.9	92.5	72.8	92.3	93.2	91.4	93.4
25 [-]	28.2k / 56	57.0	44.8	48.2	49.1	54.6	47.2	49.8	54.2	45.1	52.4
16 [med]	30.4k / 18	57.2	70.4	77.4	73.5	61.8	54.2	58.4	58.1	52.5	58.3
23 [med]	47.0k / 23	24.5	27.2	26.5	28.5	30.5	27.3	32.0	24.4	29.0	29.8
17 [med]	54.4k / 26	39.9	40.3	41.6	38.0	37.1	36.6	35.2	35.4	31.3	33.7
8 [IT]	61.4k / 214	46.9	53.1	55.8	53.6	48.9	45.1	48.8	50.9	43.0	46.7
1 [-]	68.1k / 122	47.2	47.5	48.7	45.1	46.8	39.1	45.4	44.2	40.7	44.9
7 [med]	91.5k / 30	41.3	35.5	41.4	39.9	41.4	36.5	37.3	37.1	40.7	41.8
11 [med]	93.0k / 38	31.6	42.6	31.8	35.4	36.0	29.6	36.7	32.7	26.5	36.6
29 [law]	109.2k / 242	65.9	69.2	67.6	67.7	66.0	63.8	65.1	64.7	62.4	65.9
27 [med]	109.3k / 49	11.0	9.6	8.7	9.2	10.0	19.4	9.4	7.9	10.7	10.6
5 [-]	109.9k / 267	46.3	47.4	46.9	45.4	44.0	42.9	43.7	44.3	40.9	45.7
6 [med]	133.4k / 73	37.2	38.9	38.7	36.8	37.5	27.5	38.0	37.2	31.3	35.9
26 [-]	134.8k / 428	31.8	30.8	31.8	31.2	31.9	32.6	32.2	30.5	29.6	31.2
15 [bank]	136.9k / 674	46.5	51.5	47.9	48.0	46.6	46.0	45.8	45.7	42.9	46.0
4 [rel]	137.4k / 1016	77.1	85.3	83.5	83.3	75.8	46.1	74.2	73.3	63.2	75.9
2 [med]	182.6k / 85	70.6	75.8	71.7	69.4	68.2	67.3	67.3	68.6	65.6	68.2
20 [med]	183.0k / 71	47.4	47.2	46.8	47.2	48.4	47.5	48.8	47.3	47.1	46.8
21 [-]	222.8k / 868	38.7	38.8	39.0	37.2	37.5	35.9	36.9	37.1	33.4	37.0
10 [med]	225.4k / 115	40.0	42.6	40.0	38.2	39.9	35.8	39.5	39.1	36.3	40.7
18 [med]	245.0k / 106	57.7	60.3	58.7	58.6	58.4	56.3	57.3	56.1	54.9	55.9
9 [med]	301.6k / 145	37.2	37.3	36.5	36.1	36.4	37.7	36.4	35.2	34.2	37.0
3 [law]	323.5k / 680	50.1	52.0	50.8	50.1	49.1	48.3	49.0	48.2	44.4	49.1
14 [med]	334.0 / 146	31.6	31.4	31.9	33.0	32.5	34.1	31.4	32.1	30.5	31.8
12 [med]	356.4k / 148	36.3	36.6	35.9	35.9	35.8	37.0	36.4	35.4	34.2	36.3

Table C.1: Complete results for the experiments with automatic domains. For each cluster, we report: the majority domain when one domain accounts for more than 75% of the class; training and test sizes; and BLEU scores obtained with the various systems used in this study. Most test sets are too small to report significance tests.

Appendix D

D - Generalized Multi-Domain Dynamic Adaptation Curriculum Algorithm (Chapter 7)

The pseudo-code for the generalized Multi-Domain Adaptation Dynamic Sampling Algorithm is in Algorithm 2.

Appendix E

E - Experiments with automatic domains (Chapter 7)

This experiment aims to simulate with automatic domains a scenario where the number of "domains" is large and where some "domains" are close and can effectively share information. Full results are in Table E.1. Cluster sizes vary from approximately 8k sentences (cluster 24) up to more than 350k sentences. More than 2/3 of these clusters mostly comprise texts from one single domain, as for cluster 12, which is predominantly MED, the remaining clusters typically mix 2 domains. According to the table, for each system, the clusters most sampled by MDAC contain most of the data of the corresponding domain. This demonstrates that MDAC is able to find related data to the task automatically. However, MDAC performance is still far behind that of the supervised scenario, as we explain in the paper.

Algorithm 2 Multi-Domain Adaptation Dynamic Sampling

Require:

- n_d corpora $C^d, d \in [1,\ldots,n_d]$ for n_d domains equipped by an empirical distribution $D_d(x)$
- n_d dev sets $Dev^d, d \in [1,\ldots,n_d]$ for n_d domains.
- Domain testing distribution $\boldsymbol{\lambda}^t \in \mathbb{R}^{n_d}$
- Batch size B
- Domain Dynamic Sampling Distribution $\lambda_i^l \in \mathbb{R}^{n_d}$ where i indexes iterations.
- $Eval\_scores = []$
- $Early\_stopping$ criterion
- Total training iterations $iter\_num$
- Update rule for sampling distribution $\boldsymbol{\lambda}_i^l$

1: **repeat**

2: // Start of iteration i

Randomly pick $d \in [1, \ldots, n_d]$ from sampling distribution λ_i^l
Sample B sentences from C^d with empirical distribution $D_d(x)$
Update model by applying SGD computed from B sampled sentences
if $i \equiv 0 \mod eval\_step$ **then**
    Evaluate current model with n_d dev sets. S_i^d is the performance at iteration i^{th} on domain d
    Report weighted score using test distribution $\boldsymbol{\lambda}^t$.

$$eval(i) = \sum_{d}^{n_d} \lambda^t(d) S_i^d$$

    $Eval\_scores.append(eval(i))$
end if
if $i \equiv 0 \mod sampler\_updating\_step$ **then**
    Update $\boldsymbol{\lambda}_i^l$
end if
if $Early\_stopping(Eval\_scores)$ **then**
    break
end if
$i = i + 1$
until $i > iter\_num$

Cl.	size	Domain content						MDAC systems					
		MED	ECB	IT	LAW	REL	TALK	MED	ECB	IT	LAW	REL	TALK
1	27436	0.24	0.11	0.47	0.17	0.00	0.01	0.01	0.00	0.01	0.00	0.00	0.01
2	68108	0.48	0.04	0.19	0.28	0.00	0.00	0.02	0.03	0.02	0.01	0.02	0.02
3	182594	1.00	0.00	0.00	0.00	0.00	0.00	0.03	0.06	0.04	0.06	0.03	0.03
4	323474	0.05	0.06	0.01	0.87	0.00	0.00	0.03	0.07	0.03	0.20	0.03	0.04
5	137451	0.03	0.00	0.00	0.00	0.86	0.10	0.03	0.04	0.04	0.06	0.17	0.04
6	109949	0.44	0.04	0.40	0.07	0.01	0.04	0.04	0.06	0.04	0.05	0.02	0.03
7	133395	0.92	0.01	0.01	0.05	0.00	0.00	0.03	0.04	0.03	0.05	0.05	0.03
8	91464	0.98	0.00	0.00	0.02	0.00	0.00	0.04	0.03	0.04	0.04	0.05	0.03
9	61353	0.02	0.01	0.96	0.01	0.00	0.00	0.03	0.02	0.03	0.01	0.04	0.02
10	301639	0.98	0.00	0.00	0.02	0.00	0.00	0.02	0.01	0.01	0.01	0.02	0.03
11	225347	0.93	0.00	0.01	0.04	0.00	0.01	0.03	0.04	0.04	0.02	0.03	0.04
12	92982	0.98	0.00	0.01	0.01	0.00	0.00	0.04	0.02	0.03	0.01	0.03	0.03
13	356377	0.99	0.00	0.00	0.01	0.00	0.00	0.03	0.03	0.04	0.03	0.03	0.04
14	17260	0.03	0.37	0.01	0.59	0.00	0.00	0.03	0.03	0.03	0.02	0.03	0.02
15	333957	0.98	0.00	0.01	0.01	0.00	0.00	0.04	0.02	0.03	0.01	0.03	0.03
16	136944	0.02	0.89	0.01	0.08	0.00	0.00	0.04	0.08	0.03	0.04	0.03	0.04
17	30443	0.96	0.01	0.02	0.01	0.00	0.00	0.04	0.03	0.03	0.03	0.04	0.03
18	54378	0.93	0.00	0.05	0.02	0.00	0.00	0.04	0.04	0.02	0.03	0.07	0.03
19	245000	0.99	0.00	0.00	0.00	0.00	0.00	0.04	0.04	0.03	0.03	0.04	0.03
20	27227	0.15	0.00	0.79	0.05	0.00	0.01	0.04	0.03	0.03	0.02	0.04	0.03
21	182990	0.99	0.00	0.00	0.01	0.00	0.00	0.03	0.02	0.03	0.03	0.03	0.03
22	222802	0.21	0.01	0.40	0.04	0.02	0.32	0.04	0.04	0.08	0.02	0.01	0.08
23	27534	0.11	0.48	0.02	0.39	0.00	0.00	0.03	0.03	0.05	0.02	0.01	0.04
24	47065	0.99	0.00	0.01	0.00	0.00	0.00	0.04	0.02	0.05	0.05	0.03	0.04
25	8129	0.95	0.00	0.04	0.01	0.00	0.00	0.03	0.03	0.04	0.03	0.02	0.02
26	28237	0.59	0.02	0.23	0.11	0.00	0.05	0.03	0.02	0.02	0.01	0.03	0.02
27	134828	0.53	0.00	0.02	0.00	0.01	0.44	0.04	0.03	0.03	0.01	0.01	0.05
28	109324	0.99	0.00	0.00	0.01	0.00	0.00	0.04	0.03	0.03	0.02	0.01	0.02
29	25561	0.56	0.16	0.08	0.16	0.00	0.04	0.03	0.03	0.03	0.02	0.02	0.03
30	109260	0.01	0.06	0.00	0.92	0.00	0.00	0.03	0.03	0.03	0.06	0.04	0.04

Table E.1: Automatic clustering experiments. We report the size of each cluster. In the 6 left columns, each line gives the proportions of the domains in each cluster. In the 6 right columns, each column corresponds to a MDAC experiment; each line gives the cumulated proportion of the corresponding cluster in the training data. For instance, when targeting the domain ECB, cluster 4 (mostly LAW) is sampled with a probability of 0.07, and cluster 16 (mostly ECB) is sampled with probability 0.08. For each system, we underline the most often sampled clusters.

Appendix F

F - Résumé en Français

Un modèle de traduction automatique neuronal (NMT) a généralement du mal à traduire des phrases dont le genre, le registre ou le thème diffèrent de ceux des phrases utilisées pour l'entraînement du modèle. Il s'agit d'une faiblesse des méthodes d'apprentissage automatique axées sur les données, dont les performances sont garanties en supposant que les distributions d'entraînement et de test sont identiques. Par conséquent, pour obtenir des performances élevées dans un domaine cible, nous devons soigneusement adapter le modèle NMT à ce domaine. Le problème de l'adaptation d'un modèle NMT à un domaine cible est appelé le **problème d'adaptation au domaine**. Deux facteurs rendent ce problème complexe, notamment la pénurie de données d'entraînement du domaine cible et le problème d'oubli catastrophique des modèles neuronaux. Le manque de données d'entraînement nous incite à exploiter des données parallèles provenant d'autres domaines pour entraîner nos modèles NMT. En effet, les modèles basés sur les réseaux neuronaux ont besoin de nombreuses données pour optimiser leurs paramètres. Par conséquent, nous devons généralement adapter notre modèle NMT au domaine cible en utilisant de nombreuses données hors domaine et une petite quantité de données du domaine cible. Deuxièmement, malgré les améliorations importantes plusieurs approches visant à adapter un modèle NMT en l'affinant à l'aide des données du domaine cible ont des performances très faibles au test hors du domaine. Ce problème est appelé **oubli catastrophique** dans la littérature de recherche sur les réseaux neuronaux. Les modèles neuronaux ont tendance à être nettement moins performants sur les données hors-domaine lorsqu'on les affine sur les données du domaine. Dans les applications réelles, nous cherchons généralement à améliorer les performances dans le domaine cible et la robustesse des modèles neuronaux par rapport aux domaines connus précédemment.

Dans cette thèse, nos contributions sont les suivantes. Premièrement, nous formalisons le problème de l'adaptation multidomaines de la traduction automatique (TA). Nous mettons en évidence quatre situations principales dans le problème de l'inadaptation des domaines. Nous fournissons une correspondance complète entre chaque situation et ses méthodes d'adaptation associées.

Ensuite, nous proposons une nouvelle évaluation multicritères pour les méthodes de

traduction automatique multidomaines. Nous réévaluons un large ensemble des méthodes avec les paramètres expérimentaux correspondant aux critères que nous proposons.

Troisièmement, nous proposons, évaluons et analysons une méthode de TA multidomaines, qui utilise des plongements lexicaux génériques conjointement avec des plongements spécifiques aux domaines. Cette méthode est beaucoup moins coûteuse que la méthode des adaptateurs résiduels qui dont l'objet d'un chapitre ultérieur. Outre une amélioration dans certains contextes multi-domaines, la méthode peut gérer un nombre croissant de domaines. Nous développons l'idée de la représentation parcimonieuse aux couches supérieures d'un modèle NMT. Nous démontrons que la performance de la méthode est équivalente à celle de plusieurs méthodes d'adaptation multidomaines. Nous proposons une nouvelle méthode d'analyse de la corrélation entre les prolongements lexicaux et le sens d'un mot, qui identifie les tokens agnostiques et spécifiques au domaine en observant la variation des K voisins les plus proches d'un mot tout en changeant son domaine.

Notre quatrième contribution est une étude approfondie de l'utilisation des adaptateurs résiduels dans l'adaptation multidomaines. Nous démontrons son efficacité et ses bonnes performances dans un contexte de la TA multidomaines, impliquant d'un grand nombre de domaines aux tailles déséquilibrées. Nous proposons différentes méthodes de régularisation pour éviter de sur-apprendre au modèle sur les domaines peu dotés. Enfin, nous proposons deux variantes plus robustes qui sont robustes par rapport aux erreurs d'étiquettes de domaine et réduisent légèrement le coût de calcul.

Ensuite, nous étudions des stratégies d'échantillonnage dynamique pour la traduction automatique multidomaines. Nous montrons que ces méthodes améliorent l'échantillonnage des données à partir du mélange des corpus par rapport à la stratégie heuristique d'échantillonnage fixe. De plus, nous démontrons leur efficacité dans plusieurs contextes particuliers tels que l'adaptation à un unique domaine, l'adaptation à un couple de domaines et l'adaptation à un domaine inconnu.

Enfin, nous étudions deux paradigmes populaires pour adapter un modèle de traduction aux domaines de test inconnus qui reposent sur la recherche de texte. Ces techniques recherchent les traductions les plus semblable et incorporent ces informations supplémentaires dans la prédiction d'un modèle NMT. Nous démontrons leur efficacité ainsi que leurs faiblesses. En outre, nous proposons une variante simple qui améliore légèrement les performances des techniques précédentes et qui est capable d'exploiter les traductions de synthèse.

Notre travail a revisité la littérature sur la traduction automatique multidomaines et a illustré plusieurs problèmes intéressants qui avaient reçu jusque là peu d'attention de la part de la communauté, comme la robustesse par rapport à des domaines de test in-

connus. Pour le cadre standard de l'adaptation multidomaines supervisée, nous avons décrit cinq propriétés d'un système multidomaines efficace. Chacune de ces exigences fondamentales ouvre de nombreuses possibilités pour améliorer la qualité d'un système multidomaines. Notre travail futur consistera à améliorer ces qualités d'un système de TA multidomaines.

Toutes les approches centrées sur les modèles partagent la même idée, qui consiste à distinguer les paramètres agnostiques au domaine et les paramètres spécifiques du domaine. Cependant, l'hétérogénéité dans la proximité des domaines laisse une question ouverte sur la possibilité d'optimiser automatiquement la partition des paramètres agnostiques et spécifiques au domaine dans un modèle MDMT plutôt que de les prédéfinir de manière heuristique. De plus, les stratégies d'échantillonnage devraient être optimisées automatiquement par rapport à la distribution du test plutôt que d'être choisies de manière heuristique. Notre étude des stratégies d'échantillonnage dynamiques est le premier pas dans cette direction. Le processus de recherche des meilleurs hyperparamètres ou des meilleures partitions pour chaque paire "problème-méthode" de l'adaptation multidomaines devrait être effectué par des approches d'apprentissage automatique telles que l'apprentissage par renforcement.

Ensuite, la TA personnalisée est un cas extrême de la TA multidomaines (Michel & Neubig, 2018) qui constitue une application intéressante dans l'industrie de la TA. Cette situation repose sur l'adaptation d'un système de TA aux styles d'écriture d'un grand nombre de traducteurs ; les systèmes de traduction multidomaines sont donc une solution possible. Cela nécessiterait cependant de développer les méthodes étudiées dans cette thèse à des milliers de domaines, ce qui reste une tâche non-triviale.

De plus, nous montrons dans un chapitre ultérieur que la TA basée sur la recherche de texte a démontré des performances étonnamment bonnes en l'adaptation multidomaines. Cependant, ce paradigme repose sur le processus de recherche de texte, qui est prédéfini a priori et sans rapport avec la tâche de traduction. Cela laisse une voie ouverte pour le développement de processus de recherche optimisé pour identifier des exemples de traduction utiles pour la phrase courante.

Enfin, nous espérons que les travaux futurs dans le domaine de la traduction automatique multidomaines s'appuieront sur nos cinq exigences axiomatiques et accorderont plus d'attention au design expérimentale afin d'évaluer avec précision la capacité de la méthode proposée. En outre, malgré l'importance des performances dans le domaine, la robustesse face à des distributions de test variables devrait être prise en compte de manière équivalente. Comme la traduction automatique multilingue, la traduction automatique aux multi-domaine est un paradigme prometteur pour l'industrie de la TA et nécessite encore beaucoup d'efforts pour atteindre ses objectifs de long terme.