Neural Machine Translation Training in a Multi-Domain Scenario

*Hassan Sajjad, Nadir Durrani, Fahim Dalvi, *Yonatan Belinkov, Stephan Vogel*

Qatar Computing Research Institute – HBKU
*MIT Computer Science and Artificial Intelligence Laboratory
{hsajjad,ndurrani,faimaduddin,svogel}@qf.org.qa, *belinkov@mit.edu

Abstract

In this paper, we explore alternative ways to train a neural machine translation system in a multi-domain scenario. We investigate data concatenation (with fine tuning), model stacking (multi-level fine tuning), data selection and multi-model ensemble. Our findings show that the best translation quality can be achieved by building an initial system on a concatenation of available out-of-domain data and then fine-tuning it on in-domain data. Model stacking works best when training begins with the furthest out-of-domain data and the model is incrementally fine-tuned with the next furthest domain and so on. Data selection did not give the best results, but can be considered as a decent compromise between training time and translation quality. A weighted ensemble of different individual models performed better than data selection. It is beneficial in a scenario when there is no time for fine-tuning an already trained model.

1. Introduction

Neural machine translation (NMT) systems are sensitive to the data they are trained on. The available parallel corpora come from various genres and have different stylistic variations and semantic ambiguities. While such data is often beneficial for a general purpose machine translation system, a problem arises when building systems for specific domains such as lectures [1, 2], patents [3] or medical text [4], where either the in-domain bilingual text does not exist or is available in small quantities.

Domain adaptation aims to preserve the identity of the in-domain data while exploiting the out-of-domain data in favor of the in-domain data and avoiding possible drift towards out-of-domain jargon and style. The most commonly used approach to train a domain-specific neural MT system is to fine-tune an existing model (trained on generic data) with the new domain [5, 6, 7, 8] or to add domain-aware tags in building a concatenated system [9]. [10] proposed a gradual fine-tuning method that starts training with complete in- and out-of-domain data and gradually reduces the out-of-domain data for next epochs. Other approaches that have been recently proposed for domain adaptation of neural machine translation are instance weighting [11, 12] and data selection [13].

In this paper we explore NMT in a multi-domain scenario. Considering a small in-domain corpus and a number of out-of-domain corpora, we target questions like:

- What are the different ways to combine multiple domains during a training process?
- What is the best strategy to build an optimal in-domain system?
- Which training strategy results in a robust system?
- Which strategy should be used to build a decent in-domain system given limited time?

To answer these, we try the following approaches: **i) data concatenation:** train a system by concatenating all the available in-domain and out-of-domain data; **ii) model stacking:** build NMT in an online fashion starting from the most distant domain, fine-tune on the closer domain and finish by fine-tuning the model on the in-domain data; **iii) data selection:** select a certain percentage of the available out-of-domain corpora that is closest to the in-domain data and use it for training the system; **iv) multi-model ensemble:** separately train models for each available domain and combine them during decoding using balanced or weighted averaging. We experiment with Arabic-English and German-English language pairs. Our results demonstrate the following findings:

- A concatenated system fine-tuned on the in-domain data achieves the most optimal in-domain system.
- Model stacking works best when starting from the furthest domain, fine-tuning on closer domains and then finally fine-tuning on the in-domain data.
- A concatenated system on all available data results in the most robust system.
- Data selection gives a decent trade-off between translation quality and training time.
- Weighted ensemble is helpful when several individual models have been already trained and there is no time for retraining/fine-tuning.

The paper is organized as follows: Section 2 describes the adaptation approaches explored in this work. We present experimental design in Section 3. Section 4 summarizes the results and Section 5 concludes.

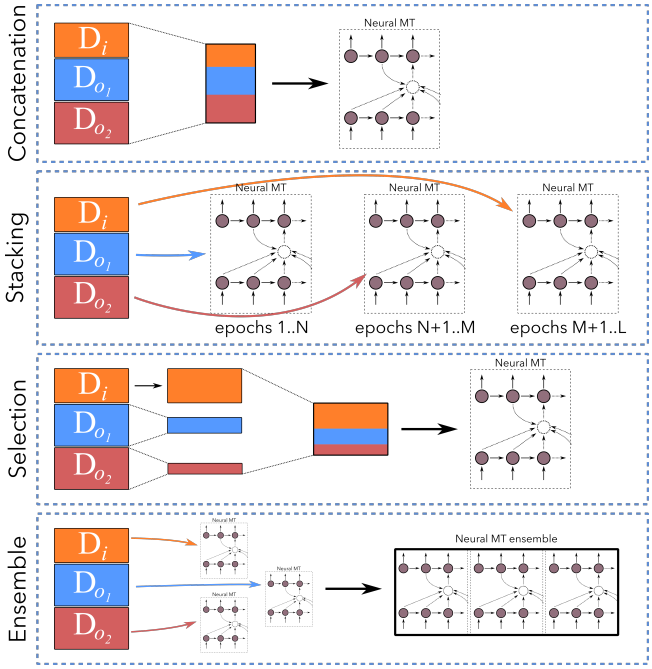

Figure 1: Multi-domain training approaches

2. Approaches

Consider an in-domain data D_i and a set of out-of-domain data $D_o = D_{o_1}, D_{o_2}, ..D_{o_n}$. We explore several methods to benefit from the available data with an aim to optimize translation quality on the in-domain data. Specifically, we try data concatenation, model stacking, data selection and ensemble. Figure 1 presents them graphically. In the following, we describe each approach briefly.

2.1. Concatenation

A naïve yet commonly used method when training both statistical [14][1] and neural machine translation systems [15] is to simply concatenate all the bilingual parallel data before training the system. During training an in-domain validation set is used to guide the training loss. The resulting system has an advantage of seeing a mix of all available data at every time interval, and is thus robust to handle heterogeneous test data.

2.2. Fine Tuning and Model Stacking

Neural machine translation follows an online training strategy. It sees only a small portion of the data in every training step and estimates the value of network parameters based on that portion. Previous work has exploited this strategy in the context of domain adaptation. [5] trained an initial model on an out-of-domain data and later extended the training on in-domain data. In this way the final model parameters are tuned towards the in-domain data. The approach is referred as *fine-tuning* later on.

Since in this work we deal with several domains, we propose a stacking method that uses multi-level fine-tuning to train a system. Figure 1 (second row) shows the complete procedure: first, the model is trained on the out-of-domain data D_{o_1} for N epochs; training is resumed from $N+1$-th epoch to the M-th epoch but using the next available out-of-domain data D_{o_2}; repeat the process till all of the available out-of-domain corpora have been used; in the last step, resume training on the in-domain data D_i for a few epochs. The resulting model has seen all of the available data as in the case of the data concatenation approach. However, here the system learns from the data domain by domain. We call this technique *model stacking*.

The model stacking and fine-tuning approaches have the advantage of seeing the in-domain data in the end of training, thus making the system parameters more optimized for the in-domain data. They also provide flexibility in extending an existing model to any new domain without having to retrain the complete system again on the available corpora.

2.3. Data Selection

Building a model, whether concatenated or stacked, on all the available data is computationally expensive. An alternative approach is *data selection*, where we select a part of the out-of-domain data which is close to the in-domain data for training. The intuition here is two fold: i) the out-of-domain data is huge and takes a lot of time to train on, and ii) not all parts of the out-of-domain data are beneficial for the in-domain data. Training only on a selected part of the out-of-domain data reduces the training time significantly while at the same time creating a model closer to the in-domain.

In this work, we use the modified Moore-Lewis [16] for data selection. It trains in- and out-of-domain n-gram models and then ranks sequences in the out-of-domain data based on cross-entropy difference. The out-of-domain sentences below a certain threshold are selected for training. Since we are dealing with several out-of-domain corpora, we apply data selection separately on each of them and build a concatenated system using in-domain plus selected out-of-domain data as shown in Figure 1. Data selection significantly reduces data size thus improving training time for NMT. However, finding the optimal threshold to filter data is a cumbersome process. Data selection using joint neural networks has been explored in [17]. We explore data selection as an alternative to the above mentioned techniques.

2.4. Multi-domain Ensemble

Out-of-domain data is generally available in larger quantity. Training a concatenated system whenever a new in-domain becomes available is expensive in terms of both time and computation. An alternative to fine-tuning the system with new in-domain is to do ensemble of the new model with the

[1] State-of-the-art baselines are trained on plain concatenation of the data with MT feature functions (such as Language Model) skewed towards in-domain data, through interpolation.

existing model. The ensemble approach brings the flexibility to use them during decoding without a need of retraining and fine-tuning.

Consider N models that we would like to use to generate translations. For each decoding step, we use the scores over the vocabulary from each of these N models and combine them by averaging. We then use these averaged scores to choose the output word(s) for each hypothesis in our beam. The intuition is to combine the knowledge of the N models to generate a translation. We refer to this approach as *balanced ensemble* later on. Since here we deal with several different domains, averaging scores of all the models equally may not result in optimum performance. We explore a variation of balanced ensemble called *weighted ensemble* that performs a weighted average of these scores, where the weights can be pre-defined or learned on a development set.

Balanced ensemble using several models of a single training run saved at different iterations has shown to improve performance by 1-2 BLEU points [15]. Here our goal is not to improve the best system but to benefit from individual models built using several domains during a single decoding process. We experiment with both balanced and weighted ensemble under the multi-domain condition only.[2]

3. Experimental Design

3.1. Data

We experiment with Arabic-English and German-English language pairs using the WIT[3] TED corpus [20] made available for IWSLT 2016 as our in-domain data. For Arabic-English, we take the UN corpus [21] and the OPUS corpus [22] as out-of-domain corpora. For German-English, we use the Europarl (EP), and the Common Crawl (CC) corpora made available for the 1^{st} Conference on Statistical Machine Translation[3] as out-of-domain corpus. We tokenize Arabic, German and English using the default *Moses* tokenizer. We did not do morphological segmentation of Arabic. Instead we apply sub-word based segmentation [23] that implicitly segment as part of the compression process.[4] Table 1 shows the data statistics after running the Moses tokenizer.

We use a concatenation of dev2010 and tst2010 sets for validation during training. Test sets tst2011 and tst2012 served as development sets to find the best model for fine-tuning and tst2013 and tst2014 are used for evaluation. We use BLEU [26] to measure performance.

3.2. System Settings

We use the Nematus tool [27] to train a 2-layered LSTM encoder-decoder with attention [28]. We use the default settings: embedding layer size: 512, hidden layer size: 1000. We limit the vocabulary to 50k words using BPE [23] with 50,000 operations.

Arabic-English			
Corpus	Sentences	Tok$_{ar}$	Tok$_{en}$
TED	229k	3.7M	4.7M
UN	18.3M	433M	494M
OPUS	22.4M	139M	195M
German-English			
Corpus	Sentences	Tok$_{de}$	Tok$_{en}$
TED	209K	4M	4.2M
EP	1.9M	51M	53M
CC	2.3M	55M	59M

Table 1: Statistics of the Arabic-English and German-English training corpora in terms of Sentences and Tokens. EP = Europarl, CC = Common Crawl, UN = United Nations.

4. Results

In this section, we empirically compare several approaches to combine in- and out-of-domain data to train an NMT system. Figure 2 and Figure 3 show the learning curve on development sets using various approaches mentioned in this work. We will go through them individually later in this section.

4.1. Individual Systems

We trained systems on each domain individually (for 10 epochs)[5] and chose the best model using the development set. We tested every model on the in-domain testsets. Table 2 shows the results. On Arabic-English, the system trained on the out-of-domain data OPUS performed the best. This is due to the large size of the corpus and its spoken nature which makes it close to TED in style and genre. However, despite the large size of UN, the system trained using UN performed poorly. The reason is the difference in genre of UN from the TED corpus where the former consists of United Nations proceedings and the latter is based on talks.

For German-English, the systems built using out-of-domain corpora performed better than the in-domain corpus. The CC corpus appeared to be very close to the TED domain. The system trained on it performed even better than the in-domain system by an average of 2 BLEU points.

4.2. Concatenation and Fine-tuning

Next we evaluated how the models performed when trained on concatenated data. We mainly tried two variations: i) concatenating all the available data (*ALL*) ii) combine only the available out-of-domain data (*OD*) and later fine-tune the

[2]Weighted fusion of Neural Networks trained on different domains has been explored in [18] for phrase-based SMT. Weighted training for Neural Network Models has been proposed in [19].

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

[4][24] showed that using BPE performs comparable to morphological tokenization [25] in Arabic-English machine translation.

[5]For German-English, we ran the models until they converged because the training data is much smaller compared to Arabic-English direction

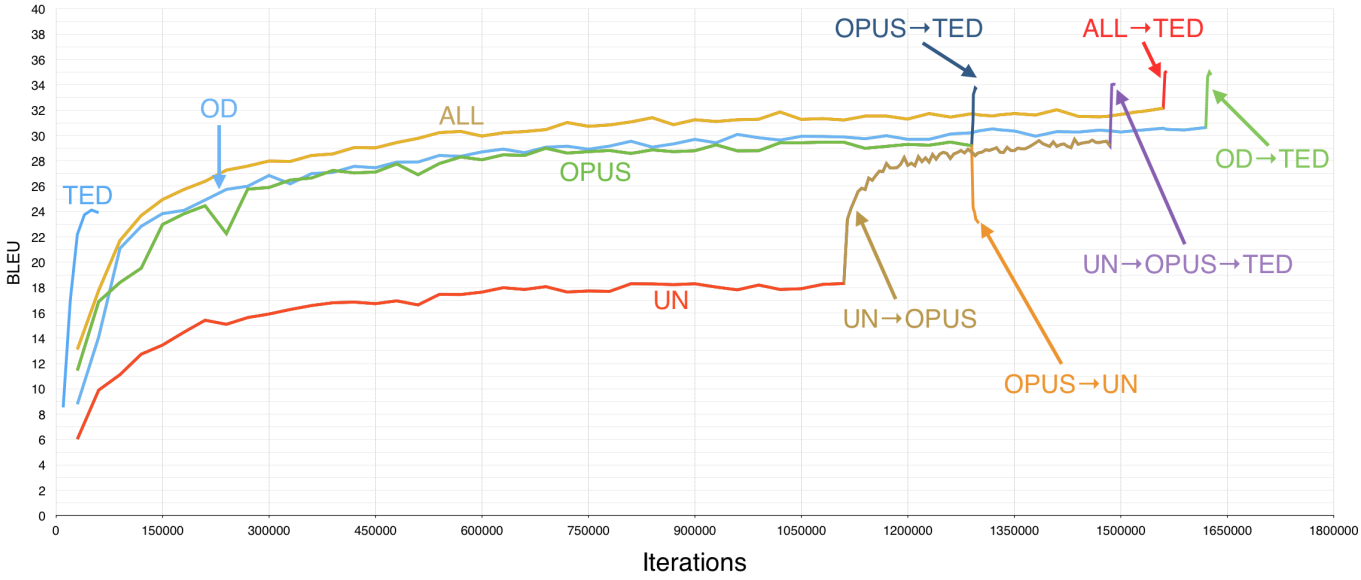

Figure 2: Arabic-English system development life line evaluated on development set tst-11 and tst-12. Here, `ALL` refers to `UN+OPUS+TED`, and `OD` refers to `UN+OPUS`

Arabic-English			
	TED	UN	OPUS
tst13	23.6	22.4	**32.2**
tst14	20.5	17.8	**27.3**
avg.	22.1	20.1	**29.7**
German-English			
	TED	CC	EP
tst13	29.5	**29.8**	29.1
tst14	23.3	**25.7**	25.1
avg.	26.4	**27.7**	27.1

Table 2: Individual domain models evaluated on TED testsets

Arabic-English				
	TED	ALL	OD→TED	ALL→TED
tst13	23.6	36.1	37.9	38.0
tst14	20.5	30.2	32.1	32.2
avg.	22.1	33.2	35.0	35.1
German-English				
	TED	ALL	OD→TED	ALL→TED
tst13	29.5	35.7	38.1	38.1
tst14	23.3	30.8	32.8	32.9
avg.	28.0	33.3	35.4	35.5

Table 3: Comparing results of systems built on a concatenation of the data. OD represents a concatenation of the out-of-domain corpora and ALL represents a concatenation of OD and the in-domain data. → sign means fine-tuning

model on the in-domain data. Table 3 shows the results. The fine-tuned system outperformed a full concatenated system by 1.8 and 2.1 average BLEU points in Arabic-English and German-English systems respectively.

Looking at the development life line of these systems (Figures 2, 3), since *ALL* has seen all of the data, it is better than *OD* till the point *OD* is fine-tuned on the in-domain corpus. Interestingly, at that point *ALL* and *OD*→TED have seen the same amount of data but the parameters of the latter model are fine-tuned towards the in-domain data. This gives it average improvements of up to 2 BLEU points over *ALL*.

The *ALL* system does not give any explicit weight to any domain [6] during training. In order to revive the in-domain data, we fine-tuned it on the in-domain data. We achieved comparable results to that of the OD→TED model which means that one can adapt an already trained model on all the available data to a specific domain by fine tuning it on the domain of interest. This can be helpful in cases where in-domain data is not known beforehand.

4.3. Model Stacking

Previously we concatenated all out-of-domain data and fine-tuned it with the in-domain TED corpus. In this approach, we picked one out-of-domain corpus at a time, trained a model and fine-tuned it with the other available domain. We repeated this process till all out-of-domain data had been used. In the last step, we fine-tuned the model on the in-domain data. Since we have a number of out-of-domain corpora available, we experimented with using them in different per-

[6] other than the data size itself

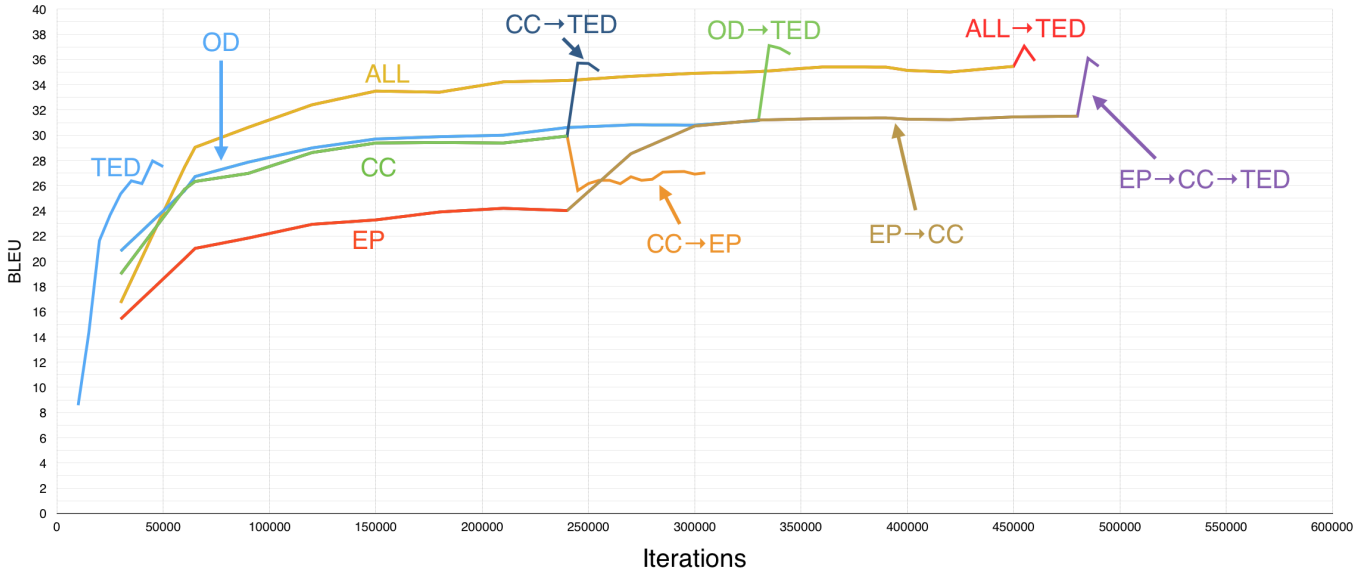

Figure 3: German-English system development life line evaluated on development set tst-11 and tst-12. Here, `ALL` refers to `EP+CC+TED`, and `OD` refers to `EP+CC`

mutations for training and analyzed their effect on the development sets. Figure 2 and Figure 3 show the results. It is interesting to see that the order of stacking has a significant effect on achieving a high quality system. The best combination for the Arabic-English language pair started with the UN data, fine-tuned on OPUS and then fine-tuned on TED. When we started with OPUS and fine-tuned the model on UN, the results dropped drastically as shown in Figure 2 (see OPUS→UN). The model started forgetting the previously used data and focused on the newly provided data which is very distant from the in-domain data. We saw similar trends in the case of German-English language pair where CC→EP dropped the performance drastically. We did not fine-tune CC→EP and OPUS→UN on TED since there was no better model to fine-tune than to completely ignore the second corpus i.e. UN and EP for Arabic and German respectively and fine-tune OPUS and CC on TED. The results of OPUS→TED and CC→TED are shown in Figures.

Comparing the OPUS→TED system with the UN→OPUS→TED system, the result of OPUS→TED are lowered by 0.62 BLEU points from the UN→OPUS→TED system. Similarly, we saw a drop of 0.4 BLEU points for German-English language pair when we did not use EP and directly fine-tuned CC on TED. There are two ways to look at these results, considering quality vs. time: i) by using UN and EP in model stacking, the model learned to remember only those parts of the data that are beneficial for achieving better translation quality on the in-domain development sets. Thus using them as part of the training pipeline is helpful for building a better system. ii) training on UN and EP is expensive. Dropping them from the pipeline significantly reduced the training time and resulted in a loss of 0.62 and 0.4 BLEU points only.

To summarize, model stacking performs best when it starts from the domain furthest from the in-domain data. In the following, we compare it with the data concatenation approach.

4.4. Stacking versus Concatenation

We compared model stacking with different forms of concatenation. In terms of data usage, all models are exposed to identical data. Table 4 shows the results. The best systems are achieved using a concatenation of all of the out-of-domain data for initial model training and then fine-tuning the trained model on the in-domain data. The concatenated system *ALL* performed the lowest among all.

ALL learned a generic model from all the available data without giving explicit weight to any particular domain whereas model stacking resulted in a specialized system for the in-domain data. In order to confirm the generalization ability of *ALL* vs. model stacking, we tested them on a new domain, News. *ALL* performed 4 BLEU points better than model stacking in translating the news NIST MT04 testset. This concludes that a concatenation system is not an optimum solution for one particular domain but is robust enough to perform well in new testing conditions.

4.5. Data Selection

Since training on large out-of-domain data is time inefficient, we selected a small portion of out-of-domain data that is closer to the in-domain data. For Arabic-English, we selected 3% and 5% from the UN and OPUS data respectively which constitutes roughly 2M sentences. For German-English, we

	Arabic-English		
	ALL	OD→TED	UN→OPUS→TED
tst13	36.1	37.9	36.8
tst14	30.2	32.1	31.2
avg.	33.2	35.0	34.0
	German-English		
	ALL	OD→TED	EP→CC→TED
tst13	35.7	38.1	36.8
tst14	30.8	32.8	31.7
avg.	33.3	35.4	34.3

Table 4: Stacking versus concatenation

	Arabic-English		German-English	
	ALL	Selected	ALL	Selected
tst13	36.1	32.7	35.7	34.1
tst14	30.2	27.8	30.8	29.9
avg.	33.2	30.3	33.3	32.0

Table 5: Results of systems trained on a concatenation of selected data and on a concatenation of all available data

selected 20% from a concatenation of EP and CC, which roughly constitutes 1M training sentences.[7]

We concatenated the selected data and the in-domain data to train an NMT system. Table 5 presents the results. The selected system is worse than the *ALL* system. This is in contrary to the results mentioned in the literature on phrase-based machine translation where data selection on UN improves translation quality [29]. This shows that NMT is not as sensitive as phrase-based to the presence of the out-of-domain data.

Data selection comes with a cost of reduced translation quality. However, the selected system is better than all individual systems shown in Table 2. Each of these out-of-domain systems take more time to train than a selected system. For example, compared to individual UN system, the selected system took approximately 1/10th of the time to train. One can look at data selected system as a decent trade-off between training time and translation quality.

4.6. Multi-domain Ensemble

We took the best model for every domain according to the average BLEU on the development sets and ensembled them during decoding. For weighted ensemble, we did a grid search and selected the weights using the development set. Table 6 presents the results of an ensemble on the Arabic-English language pair and compares them with the individual best model, OPUS, and a model built on *ALL*. As expected,

[7]These data-selection percentages have been previously found to be optimal when training phrase-based systems using the same data. For example see [29].

	Arabic-English			
	OPUS	ALL	ENS_b	ENS_w
tst13	32.2	36.1	31.9	34.3
tst14	27.3	30.2	25.8	28.6
avg.	29.7	33.2	28.9	31.5

Table 6: Comparing results of balanced ensemble (ENS_b) and weighted ensemble (ENS_w) with the best individual model and the concatenated model

balanced ensemble (*ENS*$_b$) dropped results compared to the best individual model. Since the domains are very distant, giving equal weights to them hurts the overall performance. The weighted ensemble (*ENS*$_w$) improved from the best individual model by 1.8 BLEU points but is still lower than the concatenated system by 1.7 BLEU points. The weighted ensemble approach is beneficial when individual domain specific models are already available for testing. Decoding with multiple models is more efficient compared to training a system from scratch on a concatenation of the entire data.

4.7. Discussion

The concatenation system showed robust behavior in translating new domains. [9] proposed a domain aware concatenated system by introducing domain tags for every domain. We trained a system using their approach and compared the results with simple concatenated system. The domain aware system performed slightly better than the concatenated system (up to 0.3 BLEU points) when tested on the in-domain TED development sets. However, domain tags bring a limitation to the model since it can only be tested on the domains it is trained on. Testing on an unknown domain would first require to find its closest domain from the set of domains the model is trained on. The system can then use that tag to translate unknown domain sentences.

5. Conclusion

We explored several approaches to train a neural machine translation system under multi-domain conditions and evaluated them based on three metrics: translation quality, training time and robustness. Our results showed that an optimum in-domain system can be built using a concatenation of the out-of-domain data and then fine-tuning it on the in-domain data. A system built on the concatenated data resulted in a generic system that is robust to new domains. Model stacking is sensitive to the order of domains it is trained on. Data selection and weighted ensemble resulted in a less optimal solution. The former is efficient to train in a short time and the latter is useful when different individual models are available for testing. It provides a mix of all domains without retraining or fine-tuning the system.

6. Acknowledgments

The research presented in this paper is partially conducted as part of the the European Unions Horizon 2020 research and innovation programme under grant agreement 644333 (SUMMA).

7. References

[1] F. Guzmán, H. Sajjad, S. Vogel, and A. Abdelali, "The AMARA corpus: Building resources for translating the web's educational content," in *Proceedings of the 10th International Workshop on Spoken Language Technology (IWSLT-13)*, December 2013.

[2] M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, and M. Federico, "Report on the 11th IWSLT Evaluation Campaign," *Proceedings of the International Workshop on Spoken Language Translation, Lake Tahoe, US*, 2014.

[3] A. Fujii, M. Utiyama, M. Yamamoto, and T. Utsuro, "Overview of the patent translation task at the ntcir-8 workshop," in *In Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access*, 2010, pp. 293–302.

[4] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna, "Findings of the 2014 workshop on statistical machine translation," in *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA, June 2014.

[5] M.-T. Luong and C. D. Manning, "Stanford Neural Machine Translation Systems for Spoken Language Domains," in *Proceedings of the International Workshop on Spoken Language Translation*, Da Nang, Vietnam, December 2015.

[6] M. Freitag and Y. Al-Onaizan, "Fast domain adaptation for neural machine translation," *CoRR*, vol. abs/1612.06897, 2016. [Online]. Available: http://arxiv.org/abs/1612.06897

[7] C. Servan, J. M. Crego, and J. Senellart, "Domain specialization: a post-training domain adaptation for neural machine translation," *CoRR*, vol. abs/1612.06141, 2016. [Online]. Available: http://arxiv.org/abs/1612.06141

[8] C. Chu, R. Dabre, and S. Kurohashi, "An empirical comparison of simple domain adaptation methods for neural machine translation," *CoRR*, vol. abs/1701.03214, 2017. [Online]. Available: http://arxiv.org/abs/1701.03214

[9] C. Kobus, J. M. Crego, and J. Senellart, "Domain control for neural machine translation," *CoRR*, vol. abs/1612.06140, 2016. [Online]. Available: http://arxiv.org/abs/1612.06140

[10] M. van der Wees, A. Bisazza, and C. Monz, "Dynamic data selection for neural machine translation," in *Proceedings of the the Conference on Empirical Methods in Natural Language Processing*, September 2017.

[11] R. Wang, M. Utiyama, L. Liu, K. Chen, and E. Sumita, "Instance weighting for neural machine translation domain adaptation," in *Proceedings of the the Conference on Empirical Methods in Natural Language Processing,*, September 2017.

[12] B. Chen, C. Cherry, G. Foster, and S. Larkin, "Cost weighting for neural machine translation domain adaptation," in *Proceedings of the First Workshop on Neural Machine Translation*, September 2017.

[13] R. Wang, A. Finch, M. Utiyama, and E. Sumita, "Sentence embedding for neural machine translation domain adaptation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, August 2017.

[14] P. Williams, R. Sennrich, M. Nadejde, M. Huck, B. Haddow, and O. Bojar, "Edinburgh's statistical machine translation systems for wmt16," in *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 399–410. [Online]. Available: http://www.aclweb.org/anthology/W16-2327

[15] R. Sennrich, B. Haddow, and A. Birch, "Edinburgh neural machine translation systems for wmt 16," in *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 371–376. [Online]. Available: http://www.aclweb.org/anthology/W16-2323

[16] A. Axelrod, X. He, and J. Gao, "Domain adaptation via pseudo in-domain data selection," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '11, Edinburgh, United Kingdom, 2011.

[17] N. Durrani, H. Sajjad, S. Joty, A. Abdelali, and S. Vogel, "Using Joint Models for Domain Adaptation in Statistical Machine Translation," in *Proceedings of the Fifteenth Machine Translation Summit (MT Summit XV)*. Florida, USA: AMTA, To Appear 2015.

[18] N. Durrani, H. Sajjad, S. Joty, and A. Abdelali, "A deep fusion model for domain adaptation in phrase-based mt," in *Proceedings of COLING*

2016, the 26th International Conference on Computational Linguistics: Technical Papers. Osaka, Japan: The COLING 2016 Organizing Committee, December 2016, pp. 3177–3187. [Online]. Available: http://aclweb.org/anthology/C16-1299

[19] S. Joty, H. Sajjad, N. Durrani, K. Al-Mannai, A. Abdelali, and S. Vogel, "How to Avoid Unwanted Pregnancies: Domain Adaptation using Neural Network Models," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, September 2015.

[20] M. Cettolo, "An Arabic-Hebrew parallel corpus of TED talks," in *Proceedings of the AMTA Workshop on Semitic Machine Translation (SeMaT)*, Austin, US-TX, November 2016.

[21] M. Ziemski, M. Junczys-Dowmunt, and B. Pouliquen, "The united nations parallel corpus v1.0," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*, 2016.

[22] P. Lison and J. Tiedemann, "Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), may 2016.

[23] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 1715–1725. [Online]. Available: http://www.aclweb.org/anthology/P16-1162

[24] H. Sajjad, F. Dalvi, N. Durrani, A. Abdelali, Y. Belinkov, and S. Vogel, "Challenging Language-Dependent Segmentation for Arabic: An Application to Machine Translation and Part-of-Speech Tagging," in *Proceedings of the Association for Computational Linguistics*, 2017.

[25] A. Abdelali, K. Darwish, N. Durrani, and H. Mubarak, "Farasa: A fast and furious segmenter for arabic," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 11–16. [Online]. Available: http://www.aclweb.org/anthology/N16-3003

[26] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA, USA, 2002.

[27] R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. L"aubli, A. V. Miceli Barone, J. Mokry, and M. Nadejde, "Nematus: a toolkit for neural machine translation," in *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain: Association for Computational Linguistics, April 2017, pp. 65–68. [Online]. Available: http://aclweb.org/anthology/E17-3017

[28] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015. [Online]. Available: http://arxiv.org/pdf/1409.0473v6.pdf

[29] H. Sajjad, F. Guzmn, P. Nakov, A. Abdelali, K. Murray, F. A. Obaidli, and S. Vogel, "QCRI at IWSLT 2013: Experiments in Arabic-English and English-Arabic spoken language translation," in *Proceedings of the 10th International Workshop on Spoken Language Technology (IWSLT-13)*, December 2013.