22 Advanced Topics 4: Adaptation Methods

In this section, we will cover methods for adapting sequence-to-sequence models to a particular type of problem. As a specific subset of these methods, we also often discuss **domain adaptation**: adapting models to a specific type of input data. While the word "domain" may imply that we want to handle data on a specific topic (e.g. medicine, law, sports), in reality this term is used in a broader sense, and also includes adapting to particular speaking styles (e.g. formal text vs. informal text). In this chapter we'll discuss adaptation techniques from the point of view of domain adaptation, and give some other examples in the following chapters.

The important point in considering domain adaptation methods is that we will usually have multiple training corpora of varying sizes from different domains $\langle \mathcal{F}_1, \mathcal{E}_1 \rangle, \langle \mathcal{F}_2, \mathcal{E}_2 \rangle, \ldots$. For example, domain number 1 may be a "general domain" corpus consisting of lots of random text from the web, while domain number 2 may be a "medical domain" corpus specifically focused on medical translation. There are several general approaches that can take advantage of these multiple heterogeneous types of data.

22.1 Ensembling

The first method, **ensembling**, consists of combining the prediction of multiple independently trained models together. In the case of adaptation to a particular problem, this may mean that we will have several models that are trained on the different data sources, and we combine them in an intelligent way.

This can be done, for example, by interpolating the probabilities of multiple models, as mentioned in Section 3:

$$P(E \mid F) = \alpha P_1(E \mid F) + (1 - \alpha) P_2(E \mid F) \tag{215}$$

where each of the models are trained on a different subset of the data. Within the context of phrase-based translation, this interpolation can also be done on a more fine-grained level, with the probabilities of individual phrases being interpolated together [3].

More methods for ensembling multiple models together will be covered extensively in the materials in Section 19, and thus we will not cover further details here.

22.2 Multi-task Learning

A second method for adaptation of models to particular domains is **multi-task learning** [1], a model training method that attempts to simultaneously learn models for multiple tasks, in the hope that some of the information learned from one of the tasks will be useful in solving the other. These "tasks" are loosely defined, and in the case of domain adaptation could be though of as "translate domain 1", "translate domain 2", etc. These techniques are easiest to understand in the context of neural networks, where the parameters specifying the hidden states allow us to learn compact representations of the salient information required for any particular task. If we perform multi-task learning, and the information needed to solve these two tasks overlap in some way, then training a single model on the two tasks could potentially result in learning better representations overall, increasing the accuracy on both tasks.

22.2.1 Multi-task Loss Functions

The simplest way of doing multi-task learning is to simply define two loss functions that we care about ℓ_1 and ℓ_2, and define our total loss as the sum of these two loss functions. Thus, the total corpus-level loss for a multi-task model will be the sum of the losses over the appropriate training corpora $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively:

$$\ell(\mathcal{C}_1, \mathcal{C}_2) = \ell_1(\mathcal{C}_1) + \ell_2(\mathcal{C}_2). \tag{216}$$

Once we have defined this loss, we can perform training as we normally do through stochastic gradient descent, calculating the loss for each of the tasks and performing parameter update appropriately.

One difficulty in multi-task learning is appropriately balancing the effects of different tasks on training. One obvious way is to manually add weighting coefficients λ for each task

$$\ell(\mathcal{C}_1, \mathcal{C}_2) = \lambda_1\ell_1(\mathcal{C}_1) + \lambda_2\ell_2(\mathcal{C}_2). \tag{217}$$

However, tuning these coefficients can be difficult. There are also methods to automatically adjust the weighting of each task, either by making the λ coefficients learnable [9], or by taking other approaches such as adjusting the gradients of each task to be approximately equal [4].

22.2.2 Task Labels

One simple and popular way to perform multi-task learning is to add a label to the input specifying the task at hand, such as the domain [7]. This can be done in different ways depending on the type of model at hand.

For example, in the log-linear models used in symbolic translation models such as phrase-based machine translation, this can be done by by adding domain-specific features to the log-linear model [6].

In neural MT, the most common way to do so is by adding a special token to the input indicating the domain of the desired outputs [10, 5].

22.3 Transfer Learning

The third method, **transfer learning** [14], is also based on learning from data for multiple tasks. Essentially, transfer learning usually consists of *transferring* knowledge learned on one task with large amounts of data to another task with smaller amounts of data. This could be viewed as a subset of multi-task learning where we mainly care about the results from only a single task.

22.3.1 Continued Training

The simplest way of doing so is to first train a model on task 1, then after training has concluded, start training on the actual task of interest task 2, which has significantly less training data. For, example using an SGD-style training algorithm, it is possible to first train on the general-domain data, then update the parameters on only the in-domain data [11]. This simple method is nonetheless effective, in that the latter part of training will be performed exclusively on the in-domain data, which allows this data to have a larger effect on the results than the general-domain data.

There are more sophisticated methods for performing this transfer. For example, it is possible to apply **regularization** to the parameters of the adapted model to try ensure that they remain close to the original model. [12] find that explicit regularization towards the original model parameters has a small positive effect, and a similar effect can be achieved by increasing the amount of dropout during fine tuning.

22.3.2 Data Selection

One simple but effective way to adapt language models or translation models to a particular domain is to select a subset of data that more closely matches the target domain, and only train the translation or language model on that data. One criterion that has proven effective in the selection of data for language models is the **log-likelihood differential** between a language model trained on the in-domain data and the data trained on general-domain data [13]. Specifically, if we have an in-domain corpus $\mathcal{E}_{\text{in}}$ and general-domain corpus $\mathcal{E}_{\text{gen}}$, then we train two language models $P_{\text{in}}(E)$ and $P_{\text{gen}}(E)$. Then for each sentence in $\mathcal{E}_{\text{gen}}$ we calculate its log-likelihood differential:

$$\text{diff}(E) = \log P_{\text{in}}(E) - \log P_{\text{gen}}(E). \tag{218}$$

This number basically tells us how much more likely the in-domain model thinks the sentence is than the general-domain model, and presumably sentences with higher differentials will be more likely to be similar to the sentences in the target domain. Finally, we select a threshold, and add all sentences in the general-domain corpus that have a differential higher than the threshold. This can also be done in a multi-lingual fashion to consider information on both sides of the translation pair [2], or using neural language models to improve generalization capability [8].

References

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. *Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS)*, 19:41, 2007.

[2] Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[3] Arianna Bisazza, Nick Ruiz, Marcello Federico, and FBK-Fondazione Bruno Kessler. Fill-up versus interpolation methods for phrase-based smt adaptation. In *Proceedings of the 2011 International Workshop on Spoken Language Translation (IWSLT)*, pages 136–143, 2011.

[4] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257*, 2017.

[5] Chenhui Chu, Raj Dabre, and Sadao Kurohashi. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint arXiv:1701.03214*, 2017.

[6] Jonathan H Clark, Alon Lavie, and Chris Dyer. One system, many domains: Open-domain statistical machine translation via feature augmentation. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*, 2012.

[7] Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 45, 2007.

[8] Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2013.

[9] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*, 3, 2017.

[10] Catherine Kobus, Josep Crego, and Jean Senellart. Domain control for neural machine translation. *arXiv preprint arXiv:1612.06140*, 2016.

[11] Minh-Thang Luong and Christopher D Manning. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the 2015 International Workshop on Spoken Language Translation (IWSLT)*, 2015.

[12] Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1489–1494, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[13] Robert C. Moore and William Lewis. Intelligent selection of language model training data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 220–224, 2010.

[14] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.