Latent Group Dropout for Multilingual and Multidomain Machine Translation

Anonymous ACL submission

Abstract

Multidomain and multilingual machine translation often rely on parameter sharing strategies, where large portions of the network are meant to capture the commonalities of the tasks at hand, while smaller parts are reserved to model the peculiarities of a language or a domain. In adapter-based approaches, these strategies are hardcoded in the network architecture, independent of the similarities between tasks. In this work, we propose a new method to better take advantage of these similarities, using a latent-variable model. We also develop new techniques to train this model end-to-end and report experimental results showing that the learned patterns are both meaningful and yield improved translation performance without any increase of the model size.

1 Introduction

Multidomain and multilingual machine translation aim to develop one single model to perform translation for multiple domains and multiple language pairs, respectively.[1] These paradigms are motivated by the compactness of the resulting translation system (Chu and Dabre, 2018; Dabre et al., 2020), the hypothetical positive knowledge transfer between similar domains (Pham et al., 2021) or between languages in the same family (Tan et al., 2019). However, having all the tasks use exactly the same model parameters can cause negative interference between unrelated tasks (Conneau et al., 2020; Wang et al., 2020b). Hence, the recent development of approaches relying on a partial sharing of the parameters, eg. using adapter layers as studied in (Houlsby et al., 2019; Bapna and Firat, 2019; Pham et al., 2020; Philip et al., 2020). If these techniques have proven effective for building strong baselines, they fail to fully take advantage of the similarities that exist between domains and tasks, as documented eg. in (Pham et al., 2021). This is because the partition of the parameter space between generic or task-specific subparts, and their allocation to each task, is hard-coded in the network, irrespective of the actual commonalties and differences in the data space.

In this work, we study and develop a new method, *multi-task group dropout*, aimed to take into account the similarity between tasks in a more effective way, by *learning the network organization from the data*. To this end, we introduce a set of latent variables in the model, to account for the unseen association between tasks and regions of the representation space and show how training can still be performed end-to-end using a variational surrogate of the log-likelihood loss function. Our experiments with multilingual and multidomain machine translation confirm that this method can automatically detect similarities in the data, meaning that related tasks use the same subparts of the network. Our results also show that this method is comparable to using adapter layers in a number of empirical comparisons; however, contrarily to adapters, these performance are obtained without any increase of the model size. Our contributions are primarily methodological and can be summarized as follows:

1. We introduce a novel, sound mathematical formulation to the problem of jointly learning task-dependent sub-networks and the parameters of the underlying models using variational probabilistic modeling techniques;
2. We present algorithms to train this model end-to-end with very little extra parameters;
3. We report, using an extensive set of experiments, gains for multidomain MT and very low-resourced languages in multilingual MT;
4. We study how this method can actually exploit the similarities between tasks to learn interpretable sub-networks.

[1] We will refer to these two situations as 'multi-task MT' and refer to individual domains and languages as 'tasks'.

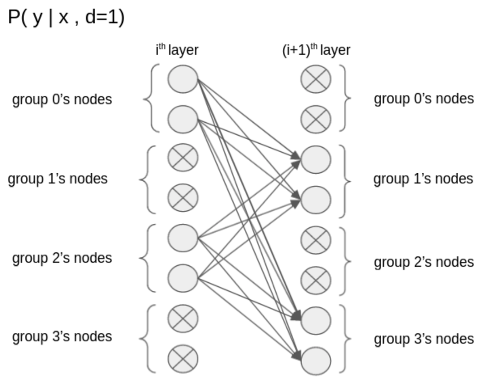


Figure 1: Latent group dropout. The set of nodes in each layer is divided into equal-sized groups. For each task, we only keep a fixed number of active groups of nodes and nullify all the other nodes.

2 Multi-task group dropout

2.1 Network architecture, groups and layers

Many architectures for multitask learning are based on a matching of subset of model parameters with tasks. Given the task and the input instance, only a subpart of the network will be involved in the computation of the output value, based on a predefined association between subnetworks and tasks. The adapter architecture of (Bapna and Firat, 2019) illustrates this strategy, where a task-dependent set of layers is activated for each task.

In our approach, we also require to know the task $d \in [0 \dots n_d - 1]$ for each training and test instance. The structure of our Transformer networks (Vaswani et al., 2017) is however based on the notion of *groups of nodes in the computation graph*. At the input of each Transformer layer $l \in [1 \dots L]$, we partition all input state vectors into n_p groups of nodes, and zero-out a task-dependent subset of these groups. The assignment of tasks to groups will be learned from the data, under the constraint that each task only activates exactly k groups of *active nodes*, while the all the other values are nullified, akin to a dropout process (see Figure 1). Formally, a *group dropout mask* m_l^d is a n_p-dimensional binary vector containing exactly k ones: group p ($\in [0, \dots, n_p$-1]) is retained for task d if $m_l^d(p) = 1$ and is dropped if $m_l^d(p) = 0$. We denote $\Delta_k^{n_p} = \{m \in \{0,1\}^{n_p}$ such that $\mid m \mid_{L_1} = k\}$ the set of all admissible masks, with $\mid m \mid_{L_1}$ the L_1 norm of vector m; $\#\Delta_k^{n_p}$ is the cardinal of Δ^{n_p}.

Given m_l^d, the mask r_l^d for task d in layer l is then derived as:

$$r_l^d(i) = m_l^d(p) \quad \text{if} \quad p \times \frac{d_k}{n_p} \leqslant i < (p+1) \times \frac{d_k}{n_p},$$

where d_k is the dimension of the hidden state. The propagation of information within the network then depends on the current task value as follows:

$$\forall l \in [0, \cdots, L-1] : \tilde{h}^l = h^l \odot r_l^d,$$
$$h^{l+1} = \text{LAYER}^{l+1}(\tilde{h}^l),$$

where $\text{LAYER}^l()$ represents all the computations in Transformer layer l, $\odot$ is element-wise product. It is applied at all positions of each layer in the encoder and in the decoder.

2.2 Training with latent dropout masks

Assuming standard notation for our translation model $P(y|x,d;\theta)$ where x, y and θ respectively refer to the input, output, and parameter vector, the latent variables $m_l^d, l \in [0, \dots, L], d \in [0, \dots, n_d - 1]$ are introduced as follows. We chose the prior distribution for m_l^d as the uniform distribution over $\Delta_k^{n_p}$: $P(m_l^d|x,d;\theta) = \text{Unif}(\Delta_k^{n_p})$; variables for each layer are independent and collectively refered to as m^d. For any (variational) distribution $Q(m^1 \dots m^{n_d};\Phi)$ with parameters $\Phi = \{\phi_l^1, ..., \phi_L^{n_d}\}$, it is well-known that the log-likelihood is lower-bounded by the so-called ELBO function (hereafter denoted ℓ), made of a summation of two terms: the *distortion* D and the *rate* R defined as follows:

$$\begin{aligned} \log P(y|x,d;\theta) \geq& \ell(x,y,d;\theta,\Phi) \\ \ell(x,y,d;\theta,\Phi) =& D(x,y,d;\theta,\Phi) - R(x,y,d;\theta,\Phi) \end{aligned} \quad (1)$$

$$D(x,y,d;\theta,\phi) = \mathbb{E}_{m^d \sim Q(m^d|d,\Phi)} \log P(y|m^d,x,d;\theta)$$
$$R(x,y,d;\theta,\phi) = \text{KL}(Q(m^d|d,\Phi)||P(m^d|x,d;\theta)),$$

where KL is the Kullback-Leibler divergence. We use $-\ell(x,y,d;\theta,\Phi)$ as our surrogate training loss, as a tractable approximation of the likelihood, and try to minimize this function in θ and Φ.

The variational distribution Q of m^d is defined independently on a layerwise basis; this means that each layer only involves a subset Φ_l^d of variational parameters. Q is computed as follows:

$$\text{Ind}^d = \{i_1, \cdots, i_k\} \sim \text{SRS}(\text{softmax}(\Phi_l^d), k)$$
$$m_l^d(i) = \mathbb{I}(i \in \text{Ind}^d),$$

where $\text{SRS}(\pi, k)$ denotes the process of sampling k times without replacement from the distribution π, and $\mathbb{I}$ is the indicator function. This modeling choice for the latent vector m_l^d is motivated by the Gumbel Top-K trick of Kool et al. (2019) that we use below. Given our choices for the prior and the variational distributions, the two terms in Eq. (1) can be computed as:

$$D(\dots) = \mathbb{E}_{m^d \sim Q(m^d|d;\Phi)} \log P(y|m^d, x, d, \theta)$$
$$= \mathbb{E}_{g^d \sim^{\text{i.i.d}} G(0,1)} \left[\log P(y|\tilde{m}^d, x, d, \theta,) \right]$$

where the generation process $G(0,1)$ is a product of independent Gumbel distributions, yielding:

$$\forall d, g^d = [g_1^d, \dots, g_L^d], \text{ with } g_l^d \in \mathbb{R}^{n_p}$$
$$\forall p, g_l^d(p) \overset{\text{i.i.d}}{\sim} \text{Gumbel}(0,1)$$
$$\text{Ind}^d = \{i_1, \cdots, i_k\} = \text{Top-k} \; \{ \; g_l^d(0) + \Phi_l^d(0), \cdots, g_l^d(n_p\text{-}1) + \Phi_l^d(n_p\text{-}1) \; \} \quad (2)$$
$$\tilde{m}_l^d(p) = \mathbb{I}(p \in Ind^d).$$

For the second term, the derivation is the following:

$$R = \text{KL}(Q(m^d|d,\Phi)||P(m^d|x,d;\theta)),$$
$$= -\sum_{l=1}^{L} \left(\mathbb{H}\left[Q(m_l^d|d,\Phi)\right] - \log(\#\Delta_k^{n_p}) \right)$$
$$= -\sum_{l=1}^{L} \left(\mathbb{H}\left[Q(i_1,\cdots,i_k|d,\Phi)\right] - \log(\#\Delta_k^{n_p}) \right)$$
$$\leqslant -\sum_{l=1}^{L} \left(\mathbb{H}\left[Q(i_1|d,\Phi_l^d)\right] - \log(\#\Delta_k^{n_p}) \right). \quad (3)$$

We prove inequality (3) in Appendix B. This inequality shows that an upperbound of R is $\sum_{l=1}^{L}(\log(\#\Delta_k^{n_p}) - \mathbb{H}(\text{softmax}(\Phi_l^d)))$ since $i_1|\Phi_l^d \sim \text{softmax}(\Phi_l^d)$. During training, we thus maximize a sum over layers of the entropy $\mathbb{H}(\text{softmax}(\Phi_l^d))$ which performs a regularization over the parameters Φ^d of the variational distribution.

Thanks to the Gumbel Top-K trick, we can move the parameters Φ into the objective function and get rid of policy gradients, which have been reported to be very unstable (Kingma and Welling, 2014). However, the operator Top-k, which serves to define $\tilde{m}_l^d$ in Equation (2), is not differentiable. Therefore, we approximate this function by the Soft-Top-K function defined as follows:

$$\hat{m}_l^d(\tau) = \underset{\substack{0 \leqslant m_i \leqslant 1 \\ \forall 0 \leqslant i \leqslant n_d\text{-}1 \\ 1^T.m = k}}{argmin} \; -(g_l^d + \Phi_l^d)^T.m - \tau H_b(m) \quad (4)$$

in which

$$H_b(m) = -\sum_i m_i \log(m_i) + (1-m_i)\log(1-m_i).$$

In Appendix A, we prove that $\lim_{\tau \to 0} \hat{m}_l^d(\tau) = \tilde{m}_l^d$. Furthermore, we also provide the computation of $\hat{m}_l^d(\tau)$ and prove that $\hat{m}_l^d(\tau)$ is a differentiable function w.r.t Φ_l^d and that its gradients can be computed using the implicit differentiation theorem. During training, we approximate $\tilde{m}_l^d$ by $\hat{m}_l^d(\tau)$ by gradually decaying the hyper-parameter τ to 0.2. The gradient of D w.r.t Φ_l^d is computed using the chain rule as follows:

$$\frac{\partial D}{\partial \Phi_l^d} = \frac{\partial D}{\partial \hat{m}_l^d(\tau)} \times \frac{\partial \hat{m}_l^d(\tau)}{\partial \Phi_l^d}$$

The gradient $\frac{\partial D}{\partial \hat{m}_l^d(\tau)}$ is computed via autograd algorithm while $\frac{\partial \hat{m}_l^d(\tau)}{\partial \Phi_l^d}$ is computed via implicit differentiation, as explained in Appendix A.

We jointly train the Transformer parameters θ and the parameters of the variational distribution Φ using the following multi-task loss.

$$\mathscr{L}(\theta,\Phi) = \sum_{d=1}^{n_d} \mathbb{E}_{x \sim \mathscr{D}_s^d, y \sim MT^d(x)} \left[-\ell(x,y,d;\theta,\Phi) \right]$$

in which $\mathscr{D}_s^d$ is distribution of task d over the input space Ω_s^d; $\text{MT}^d : \Omega_s^d \to \Omega_t^d$ is the translation function for task d, which our multi-task model needs to learn; $-\ell(x,y,d,\theta,\Phi)$ is the ELBO loss, defined in Equation 1.

Finally, during inference, we define the dropout mask for layer l and task d as follows:

$$Ind_l^d = \text{Top-k}(\Phi_l^d)$$
$$m_l^d = \mathbb{I}(i \in Ind_l^d)$$

meaning that we simply pick the k most likely parameter groups for the task at hand, and define the state dropout mask accordingly.

3 Experimental settings

3.1 System design and configuration

3.1.1 Multidomain translation systems

Our systems for the multidomain experiments are designed as follows:

- Transformer: The embedding dimension for both encoder and decoder is set as 512, and the feedforward dimension is 2048; the multi-head attention mechanism contains 8 heads; 6 layers in the encoder; 6 layers in the decoder.
- Adapter-based Transformer: The intermediate feedforward dimension is set to 2048;
- Transformer using Latent multi-task group dropout (`LaMGD` Transformers): There is no change in the architecture. We group the 512 nodes in each layer into 16 groups of 32 consecutive nodes. For each domain, only 12 out of the 16 groups are selected. The number of parameters of the variational distribution is

$L \times k \times L \times n_d$, which is negligible in comparison to the size of the Transformer model.

- Transformer using heuristic multi-task group dropout (HMGD Transformer): we share 320 nodes for every task, and reserve 32 nodes for each task (totalling $320 + 32 * 6 = 512$ nodes).

We set the dropout probability to 0.1. We train the multidomain Transformer model for 200k iterations with a batch size of 12k tokens using 4 V100 GPUs. The convergence of the standard Transformer is before 200K as its validation curve became flat near the 200K-th iteration. The LaMGD Transformer converged after 300k iterations with the same batch size. The convergence of LaMGD is controlled by its validation curve. Finally, we plug adapters to the multidomain Transformer model and finetune them for 25k iterations using the same batch size as the baseline.

3.1.2 Multilingual translation systems

The systems used in our multilingual experiments are implemented as follows:

- Multilingual Transformer: the embedding dimension for both encoder and decoder is set as 512, and the feedforward dimension is 1024, each multi-head attentions contains 8 heads as in (Wang et al., 2020a).
- Adapter based Transformer: the intermediate feedforward dimension is set as 128 as in (Gong et al., 2021a).
- LaMGD Transformer: There is no change in the architecture. We group 512 nodes in each layer into 16 groups of 32 consecutive nodes. For each language, we select 12 groups.

We set the dropout probability to 0.3. We train the multilingual Transformer model for 40k iterations with a batch size of 9600 tokens on 16 V100 GPUs as in Gong et al. (2021a). We train LaMGD Transformer for 50k iterations with the same batch size. The convergence of the models are controlled via their validation curves. Finally, we finetune the language-specific Adapters for 5k iterations.

All the translation systems are implemented with OpenNMT-tf [2] (Klein et al., 2017).

3.1.3 Hyper-parameters

We choose $n_d = 16$ so that the size of the dropout group is neither too small nor too large. The second important hyper-parameter in LaMGD is the number of selected groups in each layer, k, which we set to 12 in every experiments. By retaining 12/16 groups, we share on average 75% active groups between two domains or languages. This design ensures that the percentage of sharing is in the same ballpark as what we obtain with adapter modules. In our future work, we intend to analyze how these choices affect the final performance of the model.

The temperature parameter τ for the Soft-Top-K operator is gradually decreased from 0.5 to 0.2 according to the following policy:

$$\tau = \min\{0.2, 0.5 * \exp^{-r*step}\},$$

in which $r = 0.0001$. While Gong et al. (2021b,a) fixed τ to be 0.2, we select an anneal policy for τ proposed by previous studies (Jang et al., 2017). Finally, we set the weight of the entropy term to 0.0001 in the training loss in every experiments.

3.1.4 Latent variables initialization

We initialize the distribution of the latent variables uniformly. More precisely, we set Φ_l^d, which generates the probability of the masks via the softmax activation function, to 0^{n_d}.

3.2 Datasets and metrics

3.2.1 Multidomain translation

We use the same data as in the recent work of Pham et al. (2021) on multidomain translation. The datasets [3] for the multidomain translation experiments are detailed in Table 1. For each domain, the size of the dev set and the test set is 1 K.

3.2.2 Multilingual translation

We evaluate our model on both one-to-many (O2M) and many-to-one (M2O) translation tasks borrowing the multilingual translation datasets from past studies. More precisely, we used:

- TED8-Related. Following the setting of Wang et al. (2020a), we use a subset of translations from Qi et al. (2018) between English and eight *related languages*.
- TED8-Diverse. The dataset consists of parallel sentences between English and eight *diverse languages* as in Wang et al. (2020a).

The languages used in the multilingual experiments are as follows (see statistics in Table 2):

- **Diverse set**: bos (Bosnian), Bulgarian (bul), French (fra), ell (Greek), hin (Hindi), Korean (kor) mkd (Macedonian), mar (Marathi);
- **Related set**: Azerbajiani (aze), Belarusian (bel), Czech (ces), Galician (glg), Portuguese

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

[3] See https://github.com/qmpham/experiments/tree/main/tacl20

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

Table 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 ($\times 10^3$).

(por), Russian (rus), Slovak (slk), Turkish (tur).

For all experiments, we report the BLEU score of Papineni et al. (2002) computed with SacreBleu (Post, 2018). Statistical significance is computed with compare-mt[4] (Neubig et al., 2019). We report significant differences at the level of $p = 0.05$.

4 Results and analyses

4.1 Multidomain translation

For these experiments, our main results are in Table 3, where we observe that the LaMGD Transformer achieves a significant improvement (+2.78) over the generic Transformer system with zero extra parameters. Moreover, LaMGD Transformer achieves performance that are equivalent on average to that of the Adapter sytems, which is finetuned and contains approximately 25M additional parameters per domain. Variational mask learned from data by LaMGD also outperforms heuristic dropout mask HMGD by 0.5 in average.

4.1.1 Fuzzy domain separation

For this experiment, we reuse proposal of Pham et al. (2021), who measure the efficiency of a multidomain NMT system exploiting the proximity between domains. It uses the same data as in the previous experiment; however, the domain LAW is now randomly split into two pseudo-domains LAW_1 and LAW_2 of equal size. A truly multidomain system should be able to automatically detect the proximity between LAW_1 and LAW_2, and there should be no significant difference between the performance of a system trained with the six original domains (including LAW) or with the seven domains (including LAW by LAW_1 and LAW_2). Pham et al. (2021) reported a large gap between the two settings when using residual adapters. We replicated this setting and report the results obtained with the LaMGD Transformer system in Table 4.

The results in Table 4 show a performance decrease for the adapter-based system when training with two pseudo-domains LAW_1 and LAW_2. In contrast, the LaMGD model obtains very stable results. In Section 4.3, we show that our algorithm in fact computes the same sub-network for LAW_1 and LAW_2, that allows a full sharing of information between these two pseudo-domains.

4.2 Multilingual translation

Results for the multilingual experiments are in Table 5. The LaMGD Transformer achieves an improvement of 0.42, 0.33, 0.32 in average over the multilingual Transformer in the O2M-related, M2O-related, M2O-diverse conditions, respectively. Significant gains are observed for languages BEL, GLG (both direction), HIN and BOS (O2M direction) which are very low-resource languages in our sets. However, LaMGD Transformer is outperformed by the multilingual Transformer and language Adapters for the O2M-diverse condition.

4.3 Similarity between dropping masks

This section compares the sub-networks learnt for each domain or language pair by computing the average similarity between the corresponding dropout masks concatenated for all the layers of the underlying model. For the multidomain experiment, we analyze the case of pseud-domain separation reported in Section 4.1.1 in Figure 2a. We see that the sub-networks for LAW_1 and LAW_2 are identical, yielding a full sharing between the corresponding training sets. Furthermore, we observe a large distance between REL and the other domains, which is expected given that REL is quite distinct from the other domains. REL only share around 75% its active groups with other domains, as would be obtained by chance in our setting (see Section 3.1.3). In Figure 4, we visualize the domains using their dropping masks concatenated and mapped to a 2d space using Principal Component Analysis (PCA).

For multilingual (TED-related) experiments,

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

Related				Diverse			
LANG	TRAIN	DEV	TEST	LANG	TRAIN	DEV	TEST
Azerbaijani	5.94k	671	903	Bosnian	5.64k	474	463
Belarusian	4.51k	248	664	Marathi	9.84k	767	1090
Galician	10.0k	682	1007	Hindi	18.79k	854	1243
Slovak	61.5k	2271	2445	Macedonian	25.33k	640	438
Turkish	182k	4045	5029	Greek	134k	3344	4433
Russian	208k	4814	5483	Bulgarian	174k	4082	5060
Portuguese	185k	4035	4855	French	192k	4320	4866
Czech	103k	3462	3831	Korean	205k	4441	5637

Table 2: Data Statistics of TED8 Datasets

Model / Domain		MED	LAW	BANK	TALK	IT	REL	AVG
Transformer	[65m]	40.3	59.5	49.8	36.4	49.0	80.0	52.5
HMGD Transformer	[+0m]	40.4	60.4	51.9	38.7	50.80	86.80	54.8
Adapter	[+151m]	39.5	61.0	53.1	37.5	49.6	91.0	55.3
LaMGD Transformer	[+0m]	40.3	**60.4**	**52.4**	**39.0**	**52.4**	**87.5**	**55.3**

Table 3: Multidomain translation experiment. Boldface denotes significant gains over Transformer ($p = 0.05$)

Model / Domain		LAW	LAW_1	LAW_2
Adapter	[+151m]	61.0	60.4 (-0.6)	60.2 (-0.8)
LaMGD Transformer	[+0m]	60.4	60.4 (=)	60.4 (=)

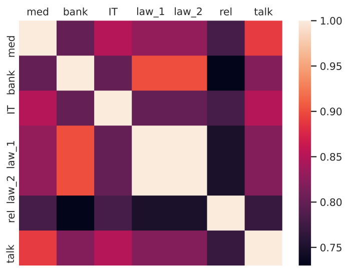
Table 4: Experiments with two similar pseudo-domains

O2M-related		AZE	BEL	CES	GLG	POR	RUS	SLK	TUR	AVG
Transformer	[91.6m]	4.8	7.3	20.8	21.1	39.7	19.8	22.6	15.2	18.9
Adapter	[+13m]	4.3	6.8	21.1	22	39.7	20	23	15.2	19
LaMGD Transformer	[+0m]	5.2	**9.4**	20.6	**22.8**	39.6	19.6	22.4	15.0	19.33
M2O-related		AZE	BEL	CES	GLG	POR	RUS	SLK	TUR	AVG
Transformer	[67.8m]	11.4	16.6	28.5	27.1	43.7	24.6	30.3	25.6	25.98
Adapter	[+13m]	10.1	15.8	28.4	26.8	43.7	24.5	30.2	25.6	25.64
LaMGD Transformer	[+0m]	11.3	**17.4**	28.6	**28.7**	43.7	24.5	30.7	25.6	26.31
O2M-diverse		BOS	MAR	HIN	MKD	ELL	BUL	FRA	KOR	AVG
Transformer	[96.9m]	10.2	4	12.7	22.2	31.8	34.0	38.3	8.3	20.19
Adapter	[+13m]	10.2	4	13.3	21.9	32.2	34.1	38.5	8.3	20.31
LaMGD Transformer	[+0m]	10.1	3.8	12.6	**22.8**	31.8	33.4	38.1	8.1	20.09
M2O-diverse		BOS	MAR	HIN	MKD	ELL	BUL	FRA	KOR	AVG
Transformer	[70.4m]	22.4	9.7	20.5	31.8	37.5	38.7	39.8	19.0	27.43
Adapter	[+13m]	22.5	9.4	20.0	30.6	37.2	38.2	39.3	19.0	27.03
LaMGD Transformer	[+0m]	**23.5**	9.6	**21.5**	32.2	37.7	38.6	40.0	18.9	27.75

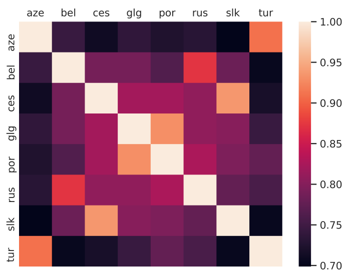
Table 5: Multilingual Translation experiments. Boldface denotes significant gains over Transformer ($p = 0.05$).

the training data contains four language families: (1) Turkic, with Azerbaijani and Turkish(AZE,TUR); (2) Slavic, with Belarusian and Russian (BEL,RUS); (3) Romance, with Galician and Portuguese (GLG, POR); and (4) Czech-Slovak, with Slovak and Czech (CES, SLK). We provide in Figure 2b the heatmap of the similarities between the dropout masks of our objective languages. We observe that each pair of languages in the same family correspond to brightest color except the diagonal in every column or every row.

We also plot the languages based on their



(a) Multidomain



(b) Multilingual (Related)

Figure 2: Heatmap visualization of the similarities between dropout masks of domains(languages).

dropout masks in Figure 3 using a 2d PCA projection.

4.4 Ablation study

We discuss here the choice of the hyper-parameters k, the number of activated nodes in each layer, and its impact on the sharing level between the tasks. Table 6 shows the variance of performance when the number of activated nodes is changed, and the sharing level between tasks decreases in consequence.

k	AVG	sharing rate
8	18.1	0.63
10	19.15	0.73
12	19.33	0.78
14	19.44	0.88

Table 6: Variation of the performance w.r.t k, while we fix $n_p = 16$ (o2m-related experiment).

5 Related work

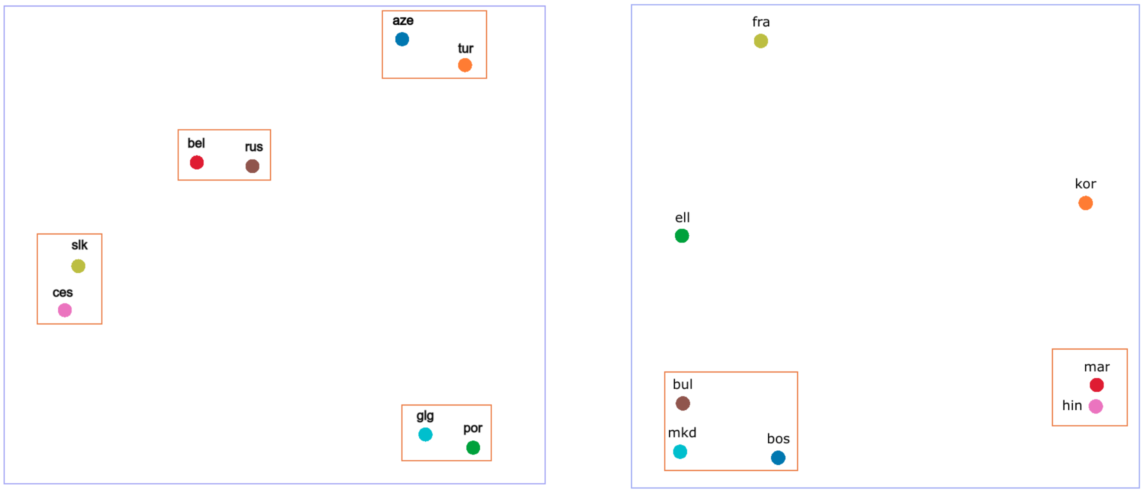
Multidomain and multilingual translation systems have received considerable attention in the recent years, and a exhaustive survey is beyond the goal of this paper. Domain adaptation for neural MT is surveyed in (Chu et al., 2017), while multidomain MT systems are notably studied in (Saunders, 2021; Pham et al., 2021); for multilingual MT, the reader is referred eg. to (Chu and Dabre, 2018; Dabre et al., 2020). We focus on the most relevant subset of this literature below.

Language similarity The methods developed by (Sen et al., 2019; Kong et al., 2021) use language proximity to design parameter sharing strategies. The authors propose a multi-decoder model sharing the same encoder among languages and routing languages in different families to different decoders. These approaches share the same interest in expressing the proximity between tasks in the selection of task-specific parameters as our approach. However, our method learn the selection from a latent commonality in data instead of using a predefined selection such as "One language family per decoder" in (Kong et al., 2021).

Language-specific sub-networks. Frankle and Carbin (2019); Liu et al. (2019) study techniques to identify the most important parameters for the current task, so that masking the less important parameters during training does not hurt performance. Lin et al. (2021) adapts this idea for multilingual NMT, trying to identify language dependent subsets of parameters by pruning a fine-tuned model. Our approach also aims to map sub-networks to tasks: we do so by masking the output of each layer, rather than masking parameters. Furthermore, Lin et al. (2021) computes the masks via a heuristic selection; while our approach learns the masks with variational techniques.

Sparse Transformer The idea of adaptive sparsity is studied in several works. For instance, Li et al. (2020) propose to use a variable depth for different tasks. The authors aimed to match the depth of the sub-network to the complexity of the task. Gong et al. (2021b,a) also take an interest in the adaptive sparse Transformers, in which differ each task triggers the selection of specific heads in multi-head attention, layers, and blocks in feedforward matrices. Mixture-of-experts (MoE) constitute another effective approach to achieve sparsity. Using the Transformer architecture, the GShard model replaces a single feedforward (FFN) sub-layer with an MoE module consisting of multiple FFN sub-layers (Lepikhin et al., 2021; Fedus et al., 2021).

Adapter modules Adapters have proven to be very efficient for multi-task NLP (Houlsby et al., 2019; Bapna and Firat, 2019; Pham et al., 2020; Pfeiffer et al., 2020). In a nutshell, this technique



(a) TED-Related

(b) TED-Diverse

Figure 3: Visualization of languages according to their dropout masks (a large vector concatenating the dropping masks of all the layers of the model) constructed by PCA.

consists in plugging several so-called adapter modules to the intermediate layers of a pretrained Transformer and finetuning these adapters on the downstream tasks. Adapters can also be trained without supervision for multilingual translation (Philip et al., 2020). However, the hard-coded separation between the domains of different tasks may lead to a catastrophic forgetting effect (Pfeiffer et al., 2021), which is a common problem in multi-task modeling using neural networks (McCloskey and Cohen, 1989). In multidomain translation, Pham et al. (2021) recently demonstrated the brittleness of adapters against fuzzy domain separations, out-of-domain distributions, and erroneous domain tags. Several subsequent studies have aimed to mitigate this weakness through a mixture of expert mechanism (e.g. (Pfeiffer et al., 2021)).

Zhang et al. (2021) propose to learn to route between shared and language-specific representations with a conditional language-specific routing while training the parameters of the underlying Transformer. This method is related to the Fusion-Adapters of Pfeiffer et al. (2021). Both approaches aim to select between shared and task-specific representations. The proximity between tasks is not taken into account in the routing mechanism. We propose a different approach to the problem of multi-task routing in the underlying network.

6 Conclusions and outlook

In this work, we have presented a novel method for multdomain and multilingual translation. It allows us to jointly search for an optimal assignment of sub-networks to tasks and to learn the parameters of the underlying network. Our method relies on a sound mathematical framework and an end-to-end optimization procedure; it only adds a small number of extra parameters. The additional training cost is also reasonable, amounting to 100k iterations in the multidomain setting, given the observed gains in performance. Experimentally, we achieve a large improvement over a Transformer baseline; our performance are also comparable to that of a strong a multi-task baseline using residual adapter modules which rely on a large number of extra parameters. For multilingual translation, our model outperforms multilingual Transformer and Language Adapters in 3 our of 4 settings. Besides, we provided an thorough analysis of the similarities between learned sub-networks and demonstrate a strong correlation between the learned similarities and the proximity of the corresponding tasks (domain or language).

There are several ways in which our methodology can be improved. In future work, we would first like to provide an complete variational framework to model both the number of groups, k and the selection of the dropout masks. Second, we also intend to dispense with the domain information during inference: this would mean replacing the dependency on d in the variational distribution by a dependency on the input x. Adressing these two questions will allow to us replace heuristic choices in the architecture design with an increased dependency on the training data.

References

Brandon Amos, Vladlen Koltun, and J. Zico Kolter. 2019. The limited multi-label projection layer. *CoRR*, abs/1906.08707.

Brandon Amos and Denis Yarats. 2020. The differentiable cross-entropy method. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 291–302. PMLR.

Ankur Bapna and Orhan Firat. 2019. 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)*, pages 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Chenhui Chu and Raj Dabre. 2018. Multilingual and multi-domain adaptation for neural machine translation. In *Proceedings of the 24st Annual Meeting of the Association for Natural Language Processing (NLP 2018)*, pages 909—912, Okayama, Japan.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391, Vancouver, Canada. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A survey of multilingual neural machine translation. *ACM Comput. Surv.*, 53(5).

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Hongyu Gong, Xian Li, and Dmitriy Genzel. 2021a. Adaptive sparse transformer for multilingual translation. *ArXiv*, abs/2104.07358.

Hongyu Gong, Yun Tang, J. Pino, and Xian Li. 2021b. Pay better attention to attention: Head selection in multilingual and multi-domain sequence modeling. *ArXiv*, abs/2106.10840.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, Long Beach, California, USA. PMLR.

Eric Jang, Shixiang Gu, and Ben Poole. 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.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72. Association for Computational Linguistics.

Xiang Kong, Adithya Renduchintala, James Cross, Yuqing Tang, Jiatao Gu, and Xian Li. 2021. Multilingual neural machine translation with deep encoder and multiple shallow decoders. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1613–1624, Online. Association for Computational Linguistics.

Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The Gumbel-top-k trick for sampling sequences without replacement. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3499–3508. PMLR.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

Xian Li, Asa Cooper Stickland, Yuqing Tang, and Xiang Kong. 2020. Deep transformers with latent depth. In *Advances in Neural Information Processing Systems*, volume 33, pages 1736–1746. Curran Associates, Inc.

Zehui Lin, Liwei Wu, Mingxuan Wang, and Lei Li. 2021. Learning language specific sub-network for multilingual machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International*

Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 293–305, Online. Association for Computational Linguistics.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019. Rethinking the value of network pruning. In *International Conference on Learning Representations*.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, 24(C):109–165.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. 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)*, pages 35–41, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Minh Quang Pham, Josep Crego, and François Yvon. 2021. Revisiting multi-domain machine translation. *Transactions of the Association for Computational Linguistics*, 9(0):17–35.

Minh Quang Pham, Josep Maria Crego, François Yvon, and Jean Senellart. 2020. A study of residual adapters for multi-domain neural machine translation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 617–628, Online. Association for Computational Linguistics.

Jerin Philip, Alexandre Berard, Matthias Gallé, and Laurent Besacier. 2020. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, Online. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535, New Orleans, Louisiana. Association for Computational Linguistics.

Danielle Saunders. 2021. Domain adaptation and multi-domain adaptation for neural machine translation: A survey. *CoRR*, abs/2104.06951.

Sukanta Sen, Kamal Kumar Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2019. Multilingual unsupervised NMT using shared encoder and language-specific decoders. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3083–3089, Florence, Italy. Association for Computational Linguistics.

Xu Tan, Jiale Chen, Di He, Yingce Xia, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with language clustering. 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)*, pages 963–973, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020a. Balancing training for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8526–8537, Online. Association for Computational Linguistics.

Zirui Wang, Zachary C. Lipton, and Yulia Tsvetkov. 2020b. On negative interference in multilingual models: Findings and a meta-learning treatment. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4438–4450, Online. Association for Computational Linguistics.

Biao Zhang, Ankur Bapna, Rico Sennrich, and Orhan Firat. 2021. Share or not? learning to schedule language-specific capacity for multilingual translation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

A Appendix A

This section explains how to compute $\hat{m}_l^d(\tau)$ by solving the optimization problem (4) and then how to compute the gradients $\frac{\partial \hat{m}_l^d(\tau)}{\partial \Phi_l^d}$.

First, to solve (4) we follow the same approach as in (Amos et al., 2019; Amos and Yarats, 2020) by applying the Karush–Kuhn–Tucker (KKT) conditions to (4). The solution of (4) will have the following form:

$$\hat{m}_l^d(\tau) = \sigma\left(\frac{g_l^d + \Phi_l^d + \bar{\nu}}{\tau}\right) \tag{5}$$

in which $\sigma(.)$ is the sigmoid function and $\bar{\nu}$ is the solution of the following equation:

$$\sum_{i=1}^{n_p} \sigma\left(\frac{g_l^d(i) + \Phi_l^d(i) + \nu}{\tau}\right) = k \tag{6}$$

Because sigmoid is monotonically increasing, equation (6) has a unique solution. Furthermore, because of the smoothness of $g(\nu, \Phi_l^d) = \sum_{i=1}^{n_p} \sigma\left(\frac{g_l^d(i) + \Phi_l^d(i) + \nu}{\tau}\right)$ w.r.t ν and Φ_l^d, we can perform the implicit differentiation of its solution $\bar{\nu}$ w.r.t Φ_l^d as below, even though the solution of (6) does not have an explicit form.

$$\frac{\partial g}{\partial \bar{\nu}} \times \frac{\partial \bar{\nu}}{\partial \Phi_l^d} + \frac{\partial g}{\partial \Phi_l^d} = 0$$

$$\Rightarrow \frac{\partial \bar{\nu}}{\partial \Phi_l^d} = -\left(\frac{\partial g}{\partial \bar{\nu}}\right)^{-1} \times \frac{\partial g}{\partial \Phi_l^d}$$

Because the differentiation of sigmoid has exact forms, $\frac{\partial g}{\partial \nu}$ and $\frac{\partial g}{\partial \Phi_l^d}$ also have exact form. Therefore, we do not need autograd to compute the implicit gradient $\frac{\partial \nu}{\partial \Phi_l^d}$. The gradient of $\hat{m}_l^d(\tau)$ w.r.t Φ_l^d is computed as follows:

$$\frac{\partial \hat{m}_l^d(\tau)}{\partial \Phi_l^d} = \frac{\partial \hat{m}_l^d(\tau)}{\partial \nu} \times \frac{\partial \nu}{\partial \Phi_l^d} + \frac{1}{\tau} \frac{exp\left(\frac{g_l^d(i) + \Phi_l^d(i) + \nu}{\tau}\right)}{\left(1 + exp\left(\frac{g_l^d(i) + \Phi_l^d(i) + \nu}{\tau}\right)\right)^2} \tag{7}$$

In our algorithm, we solve (6) by binary search. The convergence of binary search is extremely fast and assured by the monotonicity of $g(\nu, \Phi_l^d)$. In our experiments, we set the search range to $[-100, 100]$.

Finally, we need prove that $\lim_{\tau \to 0} \hat{m}_l^d(\tau) = \tilde{m}_l^d$. We assume $g_l^d(i_1) + \Phi_l^d(i_1) > g_l^d(i_2) + \Phi_l^d(i_2) > \cdots > g_l^d(i_{n_p}) + \Phi_l^d(i_{n_p})$.

Because:

$$\lim_{\tau \to 0} \sigma\left(\frac{g_l^d(i) + \Phi_l^d(i) + \nu}{\tau}\right) = \begin{cases} 1, & \text{if } \tau > -(g_l^d(i) + \Phi_l^d(i)), \\ 0, & \text{if } \tau < -(g_l^d(i) + \Phi_l^d(i)), \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

and

$$\sum_{i=1}^{n_p} \sigma\left(\frac{g_l^d(i) + \Phi_l^d(i) + \nu}{\tau}\right) = k$$

there exist ε such that $\forall \tau < \varepsilon$, the solution $\bar{\nu}$ of (6) satisfies $-(g_l^d(i_{k+1}) + \Phi_l^d(i_{k+1})) > \bar{\nu} > -(g_l^d(i_k) + \Phi_l^d(i_k))$. Furthermore, because sigmoid is monotonically increasing,

$$\sigma\left(\frac{g_l^d(i) + \Phi_l^d(i) - (g_l^d(i_k) + \Phi_l^d(i_k))}{\tau}\right) < \hat{m}_l^d(\tau)(i)$$

$$< \sigma\left(\frac{g_l^d(i) + \Phi_l^d(i) - (g_l^d(i_{k+1}) + \Phi_l^d(i_{k+1}))}{\tau}\right)$$

By taking the limit on both sides, we get the following results:

$$\lim_{\tau \to 0} \hat{m}_l^d(\tau)(i_u) = \begin{cases} 1, & \text{if } u > k \\ 0, & \text{if } u < k \end{cases}$$

And, because $\sum_{u=1}^{n_p} \hat{m}_l^d(\tau)(i_u) = k$, by taking the limit on both sides, we will have $\lim_{\tau \to 0} \hat{m}_l^d(\tau)(i_k) = 1$. Finally, we have

$$\lim_{\tau \to 0} \hat{m}_l^d(\tau)(i_u) = \begin{cases} 1, & \text{if } u \geqslant k \\ 0, & \text{if } u < k \end{cases}$$

which is equivalent to $\lim_{\tau \to 0} \hat{m}_l^d(\tau) = \tilde{m}_l^d$.

B Appendix B

In this section, we give a simple proof of inequality (3). In fact, we only need to prove $\mathbb{H}\left[P(i_1, \cdots, i_k | \Phi_l^d)\right] \geqslant \mathbb{H}\left[P(i_1 | \Phi_l^d)\right]$. The proof is as follows:

$$\mathbb{H}\left[P(i_1, \cdots, i_k | \Phi_l^d)\right] = -\mathop{\mathbb{E}}_{i_1, \cdots, i_k | \Phi_l^d}\left[\log P(i_1, \cdots, i_k | \Phi_l^d)\right]$$

$$= -\mathop{\mathbb{E}}_{i_1, \cdots, i_k | \Phi_l^d}\left[\sum_{j=2}^{k} \log P(i_j | i_1, \cdots, j_{j-1}, \Phi_l^d) + \log P(i_1 | \Phi_l^d)\right]$$

$$\geqslant -\mathop{\mathbb{E}}_{i_1, \cdots, i_k | \Phi_l^d}\left[\log P(i_1 | \Phi_l^d)\right]$$

$$= -\mathop{\mathbb{E}}_{i_1 | \Phi_l^d}\left[\log P(i_1 | \Phi_l^d)\right] = \mathbb{H}\left[P(i_1 | \Phi_l^d)\right]$$

C Appendix C

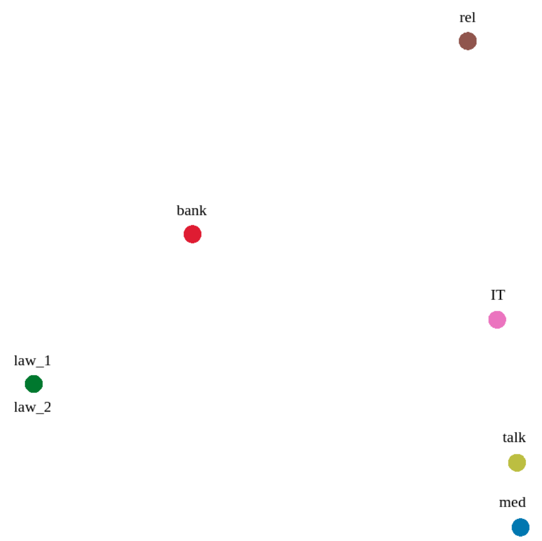


Figure 4: Visualization of domains according to their dropout masks (a large vector concatenating the dropping masks of all the layers of the model) constructed by PCA.

D Appendix D

Algorithm 1 Training `LaMGD`

Require:
- n_d corpora $C^d, d \in [1, \dots, n_d]$ for n_d domains equiped by an empirical distribution $D_d(x)$
- number of groups: n_p; number of retained groups: k
- $i = 0$; $iter\_num$

repeat

Pick a batch from domain d

Sample $\forall l, \forall p : g_l^d(p) \overset{\text{i.i.d}}{\sim} \text{Gumbel}(0,1)$

Solve the equation $\forall l$

$$\sum_{i=1}^{n_p} \sigma\left(\frac{g_l^d(i) + \Phi_l^d(i) + v}{\tau}\right) = k$$

using binary search

Compute mask of each layer

$$\forall l, \hat{m}_l^d(\tau) = \sigma\left(\frac{g_l^d + \Phi_l^d + \bar{v}}{\tau}\right)$$

Apply masks to their corresponding layer

$$\forall l \in [0, \cdots, L-1] : \tilde{h}^l = h^l \odot r_l^d,$$
$$h^{l+1} = \text{LAYER}^{l+1}(\tilde{h}^l),$$

Compute gradient of training loss over the underlying Transformer

$$\Delta_\theta = \frac{\partial L}{\partial \theta}$$

Compute gradient over the Soft-Top-K masks

$$\frac{\partial D}{\partial \hat{m}_l^d(\tau)}$$

Compute implicit gradient of the Soft-Top-K masks over Φ_l^d

$$\frac{\partial \bar{v}}{\partial \Phi_l^d} = -\left(\frac{\partial g}{\partial \bar{v}}\right)^{-1} \times \frac{\partial g}{\partial \Phi_l^d}$$

$$\frac{\partial \hat{m}_l^d(\tau)}{\partial \Phi_l^d} = \frac{\partial \hat{m}_l^d(\tau)}{\partial v} \times \frac{\partial v}{\partial \Phi_l^d} + \frac{1}{\tau} \frac{exp(\frac{g_l^d(i)+\Phi_l^d(i)+v}{\tau})}{(1 + exp(\frac{g_l^d(i)+\Phi_l^d(i)+v}{\tau}))^2}$$

Compute the gradient the training over Φ_l^d

$$\Delta_{\Phi_l^d} = \frac{\partial D}{\partial \hat{m}_l^d(\tau)} \times \frac{\partial \hat{m}_l^d(\tau)}{\partial \Phi_l^d} + \frac{\partial \mathbb{H}\left[\text{softmax}(\Phi_l^d)\right]}{\partial \Phi_l^d}$$

Update θ and Φ_l^d with their gradients

$i = i + 1$

until $i > iter\_num$