Paper Reading

Haoran Yang
2021/06/24

Content

- Self-supervised Regularization for text classification (TACL 2021)
  - Strategies of fine-tuning a pretrained LM on downstream tasks to achieve better results.

- Continual Learning for text classification with Information Disentanglement Based Regularization (NAACL 2021)

Self-supervised Regularization for Text Classification

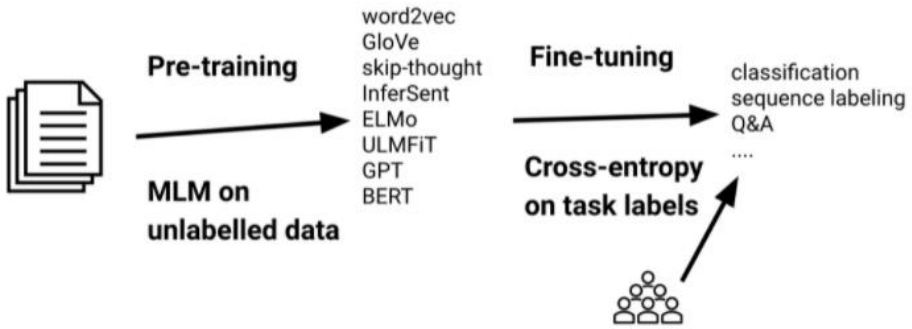
Meng Zhou[*]
Shanghai Jiao Tong University
zhoumeng9904@sjtu.edu.cn

Zechen Li [*]
Northeastern University
li.zec@northeastern.edu

Pengtao Xie[†]
UC San Diego
p1xie@eng.ucsd.edu

Background

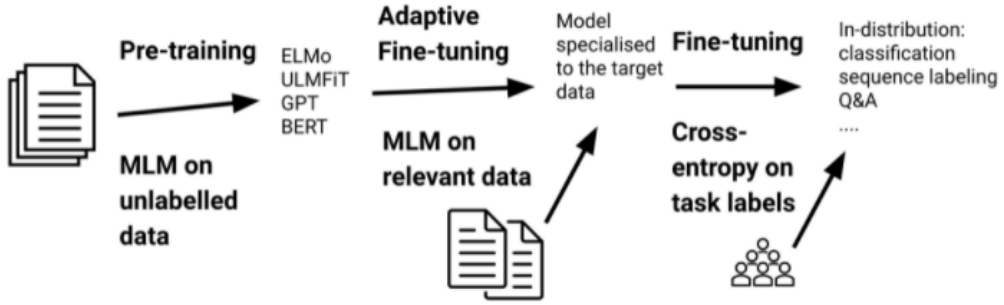
General fine-tuning

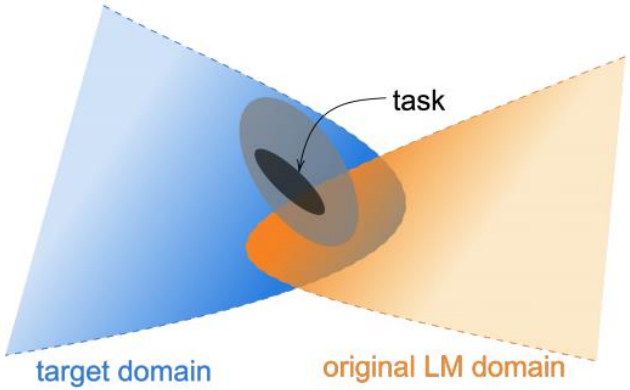


The standard pre-training—fine-tuning setting (adapted from (Ruder et al., 2019))

Pretrained LMs are still poorly equipped to deal with data that is substantially different from the one they have been pre-trained on. Adaptive fine-tuning is a way to bridge such a shift in distribution by fine-tuning the model on data that is closer to the distribution of the target data.

Adaptive fine-tuning





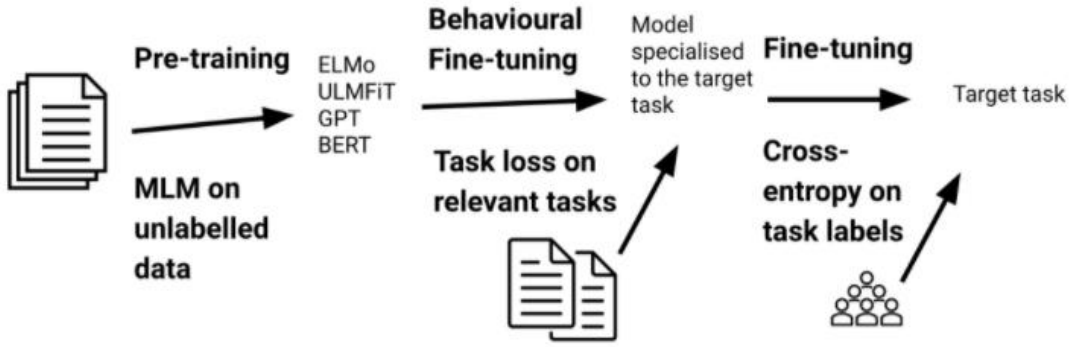
Domain-adaptive pretraining (DAPT): continue pretraining LM on a large corpus of unlabeled domain-specific text

Task-adaptive pretraining (TAPT): pretraining on the task dataset itself

Don't Stop Pretraining: Adapt Language Models to Domains and Tasks ACL 2020

Background

Behavioural fine-tuning



we can teach a model capabilities useful for doing well on the target task by fine-tuning it on relevant tasks.

For example, if we plan to fine-tune SST2 (Stanford Sentiment Treebank v2), we can firstly fine-tune a LM on IMDB dataset. (Both IMDB and SST2 are sentiment classification dataset and composed of movie reviews.)

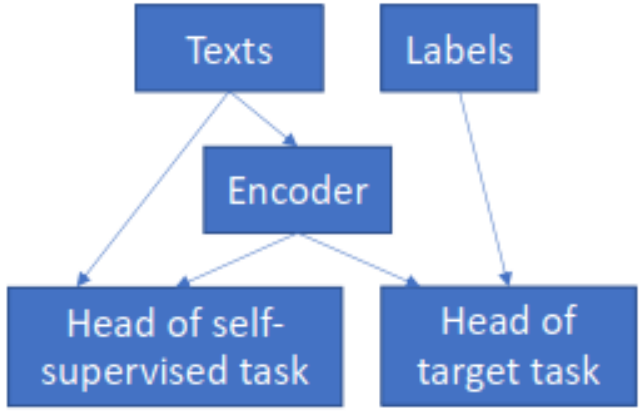
Problem of the above three fine-tuning schemes

In many real-world text classification problems, texts available for training are oftentimes limited, models tend to overfit to training data and perform less well on test data.

Proposed Method

- add data-dependent regularization losses
  - Losses are designed based on self-supervised learning so no human-labeled data is required.
  - supervised classification task and an unsupervised SSL task are performed simultaneously.

- self-supervised losses used
  - Masked Token Prediction
  - Sentence Augmentation Type Prediction
    - synonym replacement
    - synonym random insertion
    - random swap
    - random deletion



Experiments

Domain	Dataset	Label Type	Train	Dev	Test	Classes
BIOMED	CHEMPROT	relation classification	4169	2427	3469	13
	RCT	abstract sent. roles	180040	30212	30135	5
CS	ACL-ARC	citation intent	1688	114	139	6
	SCIERC	relation classification	3219	455	974	7
NEWS	HYPERPARTISAN	partisanship	515	65	65	2
	AGNEWS	topic	115000	5000	7600	4
REVIEWS	HELPFULNESS	review helpfulness	115251	5000	25000	2
	IMDB	review sentiment	20000	5000	25000	2

Dataset	RoBERTa	DAPT	TAPT	SSL-Reg	TAPT+SSL-Reg	DAPT+SSL-Reg
CHEMPROT	$81.9_{1.0}$	$84.2_{0.2}$	$82.6_{0.4}$	$83.1_{0.5}$	$83.5_{0.1}$	$\mathbf{84.4}_{0.3}$
RCT	$87.2_{0.1}$	$87.6_{0.1}$	$\mathbf{87.7}_{0.1}$	$87.4_{0.1}$	$\mathbf{87.7}_{0.1}$	$\mathbf{87.7}_{0.1}$
ACL-ARC	$63.0_{5.8}$	$75.4_{2.5}$	$67.4_{1.8}$	$69.3_{4.9}$	$68.1_{2.0}$	$\mathbf{75.7}_{1.4}$
SCIERC	$77.3_{1.9}$	$80.8_{1.5}$	$79.3_{1.5}$	$81.4_{0.8}$	$80.4_{0.6}$	$\mathbf{82.3}_{0.8}$
HYPERPARTISAN	$86.6_{0.9}$	$88.2_{5.9}$	$90.4_{5.2}$	$92.3_{1.4}$	$\mathbf{93.2}_{1.8}$	$90.7_{3.2}$
AGNEWS	$93.9_{0.2}$	$93.9_{0.2}$	$\mathbf{94.5}_{0.1}$	$94.2_{0.1}$	$94.4_{0.1}$	$94.0_{0.1}$
HELPFULNESS	$65.1_{3.4}$	$66.5_{1.4}$	$68.5_{1.9}$	$69.4_{0.2}$	$\mathbf{71.0}_{1.0}$	$68.3_{1.4}$
IMDB	$95.0_{0.2}$	$95.4_{0.1}$	$95.5_{0.1}$	$95.7_{0.1}$	$\mathbf{96.1}_{0.1}$	$95.4_{0.1}$

Unregularized RoBERTa: directly fine-tune pretrained RoBERTa on target dataset.

Task adaptive pretraining (TAPT): given the pretrained RoBERTa, it is further pretrained on the target dataset.

Domain adaptive pretraining (DAPT): given the pretrained RoBERTa, it is further pretrained on a large-size corpus whose domain is similar to the target dataset.

- we can observe:
  - Model with SSL-Reg can generally achieve better performance.
  - Model with SSL-Reg has an obvious improvement only on SCIERC and HYPERPARTISAN and HLEPFULNESS.
  - It is questionable whether the auxiliary losses can prevent overfitting. Because in low-resource data, their model only has a small improvement.

Continual Learning for Text Classification with Information Disentanglement Based Regularization

Yufan Huang[*], **Yanzhe Zhang**[*1], **Jiaao Chen, Xuezhi Wang**[2], **Diyi Yang**
Georgia Institute of Technology, [1]Zhejiang University, [2]Google
{yhuang704, jiaaochen, dyang888}@gatech.edu
[1]z_yanzhe@zju.edu.cn, [2]xuezhiw@google.com

Continual Learning

- Continual learning (CL) aims to enable information systems to learn from a data sequence across time.
  - Class incremental learning: firstly learn to classify birds and dogs, then learn to classify airplanes and cats.
  - Task incremental learning: firstly learn to **book flight** and then learn to **book hotels**, then learn to **reserve restaurants**.
  - Continually update the knowledge base during conversation, much more difficult
  - …

- Why need continual learning?
  - Most of the time, we do not want to train a separate model when a new dataset comes.
  - Multi-task training may be a solution, but need to train the model from the scratch.
  - … more human-like

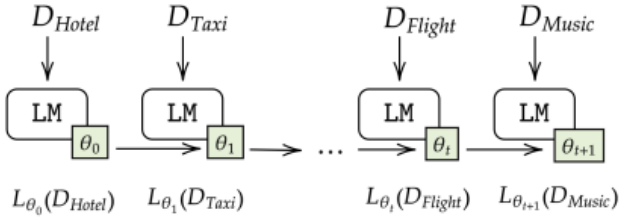


Figure 1: In Continual Learning, the model is trained one dataset at the time. In this instance, the model is first trained on data from the hotel domain D_{Hotel} then on the Taxi D_{Taxi} and so on. The parameter of the model is updated sequentially based on the loss function L.

Challenges in Continual Learning

- Not forget what it has learned from previous tasks.
  - Need the model to perform well on previous tasks
  - Overcome catastrophic forgetting

- Forward the knowledge learned in the past to help learn current task. (forward transfer)

- Transfer the knowledge backward to improve the performance of previous tasks. (backward transfer)

Problem Formulation

Given a sequence of text classification tasks $\{T_1, T_2, \ldots, T_n\}$, corresponding dataset $\{S_1, S_2, \ldots, S_n\}$, we aim to learn a model f_θ. θ is a set of parameters shared by all tasks.

$$R(f_\theta) = \sum_{i=1}^{n} \mathbb{E}_{(x^i, y^i) \sim T_i} \mathcal{L}(f_\theta(x^i), y^i)$$

Proposed Method

- **Disentangle** text hidden spaces into representations that are generic to all tasks and representations specific to each individual task
  - For task-generic information: next sentence prediction loss
  - For task-specific information: task identifier prediction loss
- **Regularization** is a constraint added to model output, hidden space and parameters to prevent parameters from changing too much while learning new tasks.

- **Replay:** a memory buffer is first adopted to store seen examples from previous tasks and then the stored data is replayed with the training set for the current task.

Proposed Method

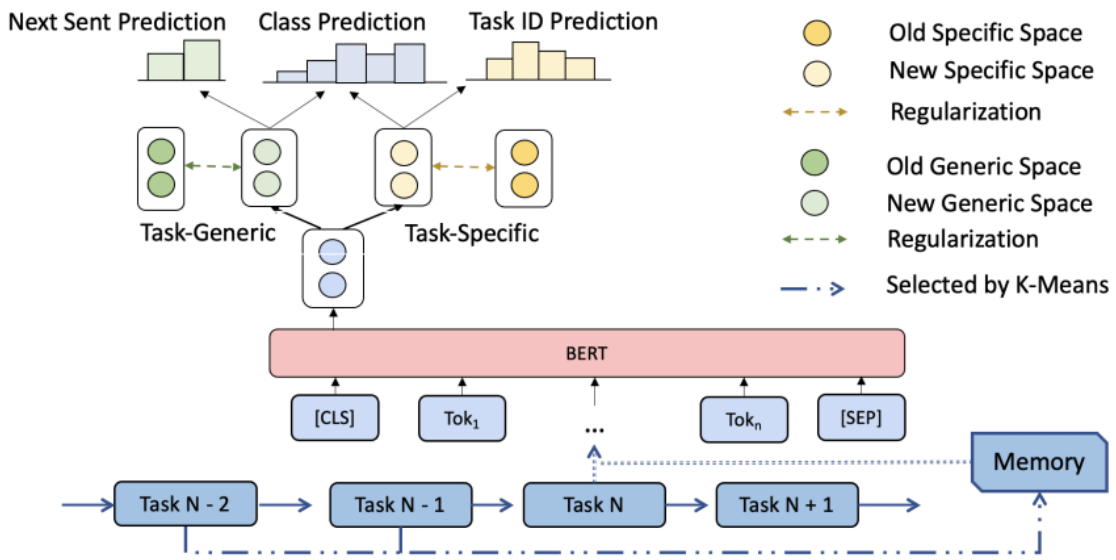


Figure 1: Our proposed model architecture. We disentangle the hidden representation into a task generic space and a task specific space via different induction biases. When training on new tasks, different spaces are regularized separately. Also, a small portion of previous data is stored and replayed.

Task-Generic space: Next sentence prediction

predictor f_{nsp} on the generic feature extractor $G(.)$

$$\mathcal{L}_{nsp} = \mathbb{E}_{x \in S_t \cup \mathcal{M}}(\mathcal{L}(f_{nsp}(G(B(x)), 0) + \mathcal{L}(f_{nsp}(G(B(\tilde{x})), 1))$$

extractor *G* needs to learn the context dependencies between two segments, which is beneficial to understand every example and generic to any individual task.

Task-specific space: Task ID prediction

i.e., distinguish which task an example belongs to.

$$\mathcal{L}_{task} = \mathbb{E}_{(x,z) \in S_t \cup \mathcal{M}} \mathcal{L}(f_{task}(S(B(x)), z)$$

where z is the corresponding task id for x.

Class Prediction: combing generic and specific features

$$\mathcal{L}_{cls} = \mathbb{E}_{(x,y) \in S_t \cup \mathcal{M}} \mathcal{L}(f_{cls}(g \circ s), y))$$

Here y is the corresponding class label for x, $f_{cls}(.)$ is the class predictor. $\circ$ denotes the concatenation of the two representations.

Proposed Method

- **Memory selection rule** After a new dataset (task) is trained, need to update the memory to store some examples of current task.
  - those stored examples should be as diverse and representative as possible.
  - using K-means to cluster $\gamma|S_t|$ clusters and only select the example closest to each cluster's centroid.

- **Regularization**

$$\mathcal{L}_{reg}^{g} = \mathbb{E}_{x \in S_t \cup \mathcal{M}_t} \| G^{t-1}(B^{t-1}(x)) - G(B(x)) \|_2$$
$$\mathcal{L}_{reg}^{s} = \mathbb{E}_{x \in S_t \cup \mathcal{M}_t} \| S^{t-1}(B^{t-1}(x)) - S(B(x)) \|_2$$

- **Overall Objective**

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{nsp} + \mathcal{L}_{task} + \lambda^{g}\mathcal{L}_{reg}^{g} + \lambda^{s}\mathcal{L}_{reg}^{s}$$

Experiments and Results

Dataset	Class	Type	Train	Test
AGNews	4	News	8000	7600
Yelp	5	Sentiment	10000	7600
Amazon	5	Sentiment	10000	7600
DBPedia	14	Wikipedia	28000	7600
Yahoo	10	Q&A	20000	7600

AGNews (news classification)
Yelp (sentiment analysis)
DBPedia (Wikipedia article classification)
Amazon (sentiment analysis)
Yahoo (Q&A classification)

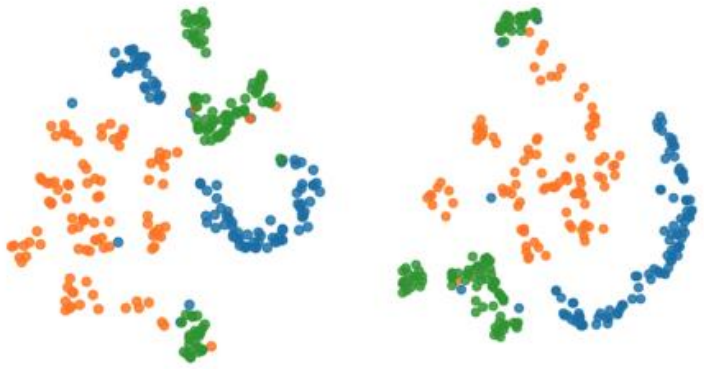
Order	Task Sequence
1	ag → yelp → yahoo
2	yelp → yahoo → ag
3	yahoo → ag → yelp
4	ag → yelp → amazon → yahoo → dbpedia
5	yelp → yahoo → amazon → dbpedia → ag
6	dbpedia → yahoo → ag → amazon → yelp
7	yelp → ag → dbpedia → amazon → yahoo

Table 2: Seven random different task sequences used for experiments. The first 6 are used in Setting (Sampled). The last 4 are used in Setting (Full).

Metric: evaluate models after training on all tasks and report their average accuracies on all test sets as the metric.

Experiments and Results

Model	Length-3 Task Sequences				Length-5 Task Sequences			
Order	**1**	**2**	**3**	**Average**	**4**	**5**	**6**	**Average**
Finetune	25.79	36.56	41.01	34.45	32.37	32.22	26.44	30.34
Replay	69.32	70.25	71.31	70.29	68.25	70.52	70.24	69.67
Regularization	71.50	70.88	72.93	71.77	72.28	73.03	72.92	72.74
IDBR	**71.80**	**72.72**	**73.08**	**72.53**	**72.63**	**73.72**	**73.23**	**73.19**
MTL	74.16	74.16	74.16	74.16	75.09	75.09	75.09	75.09



(a) Task Generic Space (b) Task Specific Space

Figure 2: t-SNE visualization of task generic hidden space and task specific hidden space of IDBR.

```
dataset_classes = {
    'amazon'  : 5,
    'yelp'    : 5,
    'yahoo'   : 10,
    'ag'      : 4,
    'dbpedia' : 14,
}

task_num = len(args.tasks)
task_classes = [dataset_classes[task] for task in args.tasks]
total_classes, offsets = compute_class_offsets(args.tasks, task_classes)
train_loaders, validation_loaders, test_loaders = \
    prepare_dataloaders(DATA_DIR, args.tasks, offsets, args.n_labeled,
                        args.n_val, args.batch_size, 16, 16)

# Reset random seed by the torch seed
np.random.seed(torch.randint(1000, [1]).item())

buffer = Memory()
model = Model(
    n_tasks=task_num,
    n_class=total_classes,
    hidden_size=args.hidden_size).to(args.device)
```

Thanks!