本文汇总几篇 text style transfer 这个任务在一些非常规场景下的模型：

1． Few shot / domain adaptive text style transfer

emnlp 19 的这篇 Domain Adaptive Text Style Transfer 希望可以用额外的数据集（source domain）来增强在目标数据集（target domain）上的表现。这篇文章所基于的基础模型仍然是基于对抗的文本风格迁移模型，包含两部分，一部分训练 AE 想要重建输入文本，一部分对于风格迁移后的文本，希望能够骗过分类器。

$$p_D(\widetilde{x}_i|c_i,\widetilde{l}_i)=\prod_{t=1}^{T}p_D(\widetilde{x}_i^t|\widetilde{x}_i^{<t},c_i,\widetilde{l}_i)$$

$$L_{ae}^{\mathcal{T}}=-\mathop{\mathbb{E}}_{x_i\sim\mathcal{T}}\log p_D(x_i|c_i,l_i)$$

$$L_{style}^{\mathcal{T}}=-\mathop{\mathbb{E}}_{\widetilde{x}_i\sim p_D(\widetilde{x}_i|c_i,\widetilde{l}_i)}\log P_{C^{\mathcal{T}}}(\widetilde{l}_i|\widetilde{x}_i)$$

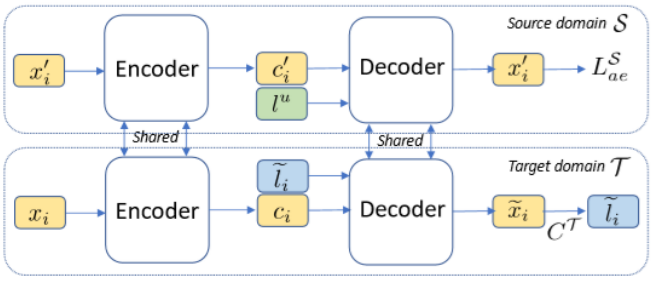
以上所示是在有 label 的目标数据集上（如 yelp）进行 transfer 训练的 loss。而如果我们还有其他数据集，则可以用来增强这个模型。这可以分为两种情况，第一种是 source domain 没有 style 的 label，仅仅是一个大数据集来增强 AE 部分的训练；第二种情况是 source domain 有 label，而且和 target domain 的 label 一致。

在第一种情况下，由于源数据集上没有 style 的 label，而训练 AE 的时候会根据不同的 style 训练不用的 decoder，或者将 style label/emb 也作为 decoder 的输入。为了在这种情况下训练 AE，本文的设计是增加一个 unknown style label，以情感转换为例，这时的 style 分为 pos、neg、unk。此时训练目标为：

$$L_{ae}^{\mathcal{S}}=-\mathop{\mathbb{E}}_{x_i'\sim\mathcal{S}}\log p_D(x_i'|c_i',l^u)$$

$$L_{\text{DAST-C}}=L_{ae}^{\mathcal{T}}+L_{style}^{\mathcal{T}}+L_{ae}^{\mathcal{S}}$$

值得注意的是，这个模型是可以应用在 few shot 的场景下的，few shot 是指只有很少的有风格的样本，比如正负各自 2k，这样的数据量是无法训练出一个好的生成模型的，但是我们可以用大量的没有 style label 的样本来进行增强。虽然该模型适用于这两种场景，但是两个场景目的是不同的，前者是希望比仅有 yelp 数据集时有更好的 transfer 效果，后者是希望在有 style 的数据不足的情况下也能学到迁移模型。

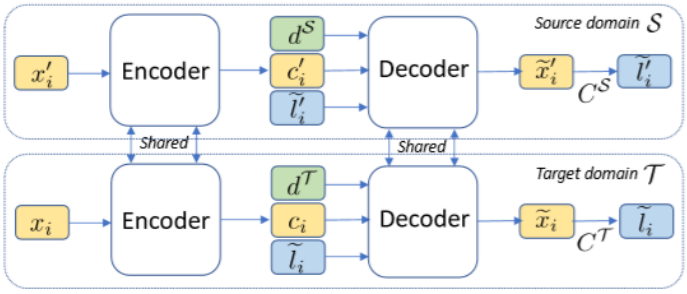


第二种情况，一个简单的方法就是将两个数据集直接混在一起，用同一个模型训练。但是由于不同的 domain 之间是有区别的，这样做无法进行 domain specific transfer，比方说同时使用 IMDB movie reviews 和 yelp restaurant reviews 训练，可能会产生这样的生成结果：the pizza is dramatic。而本文采取的方法是将一个可被训练的 domain vector 作为 decoder 的输入：

$$L_{ae}^{\mathcal{S},\mathcal{T}} = -\mathop{\mathbb{E}}_{x'_i\sim\mathcal{S}} \log p_D(x'_i|c'_i, d^{\mathcal{S}}, l'_i) - \mathop{\mathbb{E}}_{x_i\sim\mathcal{T}} \log p_D(x_i|c_i, d^{\mathcal{T}}, l_i),$$

注意到两个 domain 的 style 空间是相同的，比方说都是包含情绪的正或者负，但是本文分别在每个 domain 训练了一个分类器:

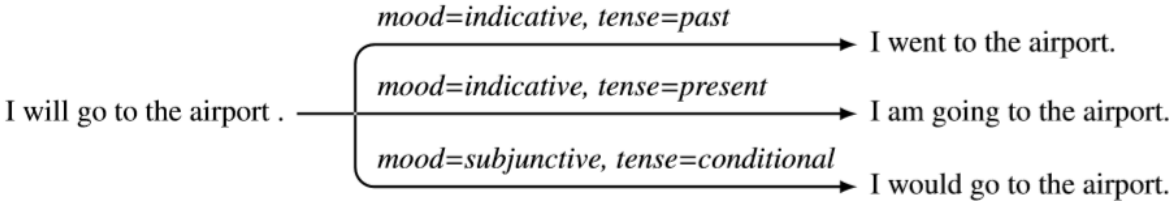
$$L_{style}^{\mathcal{S},\mathcal{T}} = -\mathop{\mathbb{E}}_{\widetilde{x}'_i\sim p_D(\widetilde{x}'_i|c'_i,d^{\mathcal{S}},\widetilde{l}'_i)} \log P_{C^{\mathcal{S}}}(\widetilde{l}'_i|\widetilde{x}'_i) - \mathop{\mathbb{E}}_{\widetilde{x}_i\sim p_D(\widetilde{x}_i|c_i,d^{\mathcal{T}},\widetilde{l}_i)} \log P_{C^{\mathcal{T}}}(\widetilde{l}_i|\widetilde{x}_i)$$



而我们还可以想到第三种场景，也就是 source domain 和 target domain 的风格不相同时，能否在 source data 上进行 target 风格的迁移，也就是 out-of-domain 的情况。

2. multiple attribute text rewriting

Nips2018 的这一篇 Content preserving text generation with attribute controls 给出了一种可以在多种属性上同时进行 transfer 的方法，对于一个数据集，如果同时有多种属性的 label，比如同时有情绪、时态、性别等，则可以训练一个模型，同时控制某生成文本的多种属性：



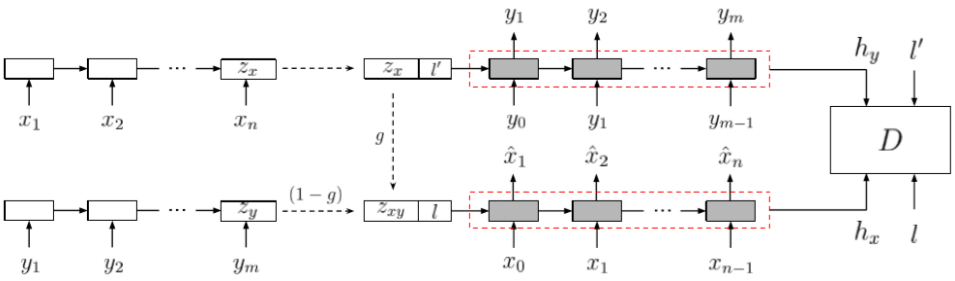
本文将多个属性的 label 处理成多个 one-hot 向量，然后拼接起来，称为 attribute vertor。本工作基本上还是一个基于对抗方法的模型，为了保持 transfer 之后的内容一致，用了以下三个 loss：

$$\mathcal{L}^{ae}(x,l) = -\log p_G(x|z_x,l)$$

$$\mathcal{L}^{bt}(x,l) = -\log p_G(x|z_y,l)$$

$$\mathcal{L}^{int} = \mathbb{E}_{(x,l)\sim p_{\text{data}},y\sim p_G(\cdot|z_x,l')}[-\log p_G(x|z_{xy},l)]$$

其中第一个 ae loss，第二个是 back-translation loss，第三个是为了避免第二个 loss 难以训练，将两者得到的隐变量进行插值，用来重建文本。



为了使得风格可以成功迁移，使用了如下对抗 loss：

$$\mathcal{L}^{\text{adv}} = \min_G \max_D \mathbb{E}_{\substack{(x,l)\sim p_{\text{data}} \\ y\sim p_G(\cdot|z_x,l')}} [2\log D(h_x,l) + \log(1-D(h_y,l')) + \log(1-D(h_x,l'))]]$$

可以看到，生成的句子不真实或者生成的句子与风格不一致都作为负例。

本文最后在同时控制句子的多种风格的任务上做了一个效果很惊艳的展示：

Mood	Tense	Voice	Neg.	john was born in the camp
Indicative	Past	Passive	No	john was born in the camp .
Indicative	Past	Passive	Yes	john wasn't born in the camp .
Indicative	Past	Active	No	john had lived in the camp .
Indicative	Past	Active	Yes	john didn't live in the camp .
Indicative	Present	Passive	No	john is born in the camp .
Indicative	Present	Passive	Yes	john isn't born in the camp .
Indicative	Present	Active	No	john has lived in the camp .
Indicative	Present	Active	Yes	john doesn't live in the camp .
Indicative	Future	Passive	No	john will be born in the camp .
Indicative	Future	Passive	Yes	john will not be born in the camp .
Indicative	Future	Active	No	john will live in the camp .
Indicative	Future	Active	Yes	john will not survive in the camp .
Subjunctive	Cond	Passive	No	john could be born in the camp .
Subjunctive	Cond	Passive	Yes	john couldn't live in the camp .
Subjunctive	Cond	Active	No	john could live in the camp .
Subjunctive	Cond	Active	Yes	john couldn't live in the camp .

在 ICLR19 上的这篇 MULTIPLE-ATTRIBUTE TEXT REWRITING 做了类似的任务，但是在这篇工作中，没有使用常见的基于对抗的方法将语义表示和风格表示进行解耦。为了探讨是否对抗的方法真的可以解耦出一个不包含风格信息的语义表示，本文先在对抗的方法上用了一个判别器之外的分类器对 encoder 的输出 z 进行分类，发现虽然判别器的分类损失会逐渐降到 50%左右，无法区分得到的表示，但是单独的分类器总是可以以百分之 80 的 acc 区分，也就是说，z 并不是一个成功剥离了风格信息的表示。

λ_{adv}	Discriminator Acc (Train)	Post-fit Classifier Acc (Test)
0	89.45%	93.8%
0.001	85.04%	92.6%
0.01	75.47%	91.3%
0.03	61.16%	93.5%
0.1	57.63%	94.5%
1.0	52.75%	86.1%
10	51.89%	85.2%
fastText	-	97.7%

因此，本文抛弃的对抗方法，只使用一个 AE 结构，其中 encoder 是将源句映射到一个隐空间，而这个空间不必像 cross align 假设的一样，会将来自不同 domain 的句子的分布对齐，而可以形成不同的分布，在 decode 的时候，附加一个风格属性作为输入，能够产生和目标 domain 一致的句子即可。训练的目标包括两项，第一项是 denoise AE，第二项同样是 back-translation loss：

$$\mathcal{L} = \lambda_{AE} \sum_{(x,y)\sim\mathcal{D}} -\log p_d\Big(x|e(x_c),y\Big) + \lambda_{BT} \sum_{(x,y)\sim\mathcal{D},\tilde{y}\sim\mathcal{Y}} -\log p_d\Big(x|e\Big(d\big(e(x),\tilde{y}\big)\Big),y\Big)$$

3. zero shot

这篇来自 facebook AI Research 的”Zero-Shot Fine-Grained Style Transfer”提出了新的场景，一是细粒度的风格属性空间，比方说用包含 24 中不同情绪的文本作为训练集，二是两种 zero shot 场景，比如在 24 种情绪中，选择 20 种来训练，然后借助一个 pretrain 的情绪分类器，可以在额外的 4 种上进行 transfer；或者在大数据量下（包含 24 种情绪的百万级数据）预训练一个模型，然后在有类似属性空间的（包含 32 种情绪的两万数据，其中部分情绪与大数据集有重合）小数据上直接做 transfer。

本文基于 multiple attribute text rewriting 提出来的模型，在这个模型训练过程中，为了给 decoder 关于风格的输入，每一种 style（pos、neg、neutral）都单独训练了一个 embedding。（以 LSTM 结构作为 decoder，风格 emb 作为 decoder 的输入，同时利用 attention 机制以 encoder 的输出作为输入。）这意味着不同的 style 有不同的实体，但是没有考虑 style 之间的相似性。

如果我们利用一些预训练的任务学到分布式的 style embedding 空间，那么不同 style 之间的相似程度可以被利用，直接用来进行 style transfer。预训练可以以分类为任务，以分类器的最后一层即包含了句子的风格信息，这一 hidden state 与类别 emb 的点积即可得到句子输入某类的概率。因此，我们可以拿出类别的 emb 用来在本任务中做风格迁移。

具体如何利用呢，举例来说，如果预训练得到的风格分布式表示是 8 维的，我们的 decoder 需要 512 维的 style 信息输入，则可用一个 512*8 的矩阵进行线性变换：$y = Wy_d$。可以对比一下原本 one-hot 的情况，原本如果有 k 种风格，则需要有 512*k 维的 embedding 矩阵，而不知道这 k 种风格彼此之间的相关性。

在 zero-shot 场景下，本模型可以直接将另外的 4 种情绪 emb 经过线性变换作为 decoder 的输入。表现如下，左边是在 20 种训练时见到过的风格上做迁移，右边是在没有见到过的四种风格上做迁移，可以看到会差很多。这个场景实际意义不大，既然有充足数据可以训练一个分类器的情况下，完全可以直接将这些风格在模型中直接训练。

Training target attribute				Held-out target attribute			
Classification				Classification			
Target	Sce	s-BL	PPL	Target	Sce	s-BL	PPL
86.8	6.0	39.5	257.2	56.5	11.6	40.2	283.9
90.5	4.2	36.7	240.5	62.2	9.2	38.5	285.5
92.6	2.8	29.7	212.4	63.4	7.5	32.3	272.6

另外一个场景，再大数据上训练之后，对于小数据集，可以直接有监督的训练，将小数据集的 32 种的情绪的 emb 映射到大数据的情绪空间，则可以不需要 fine-tune 得到 transfer，但是如下结果展示，还是在小数据上 fine-tune 比较好。总体来说，这个 zero-shot 的模型也意义不大。

	Classification			
	Target	Source	self-BLEU	PPL
Identity	1.4	56.5	100.0	96.6
Target attr. sample	77.8	0.7	0.0	94.8
Scratch	29.1	2.6	0.7	35.8
Zero-shot	3.6	30.2	62.0	135.6
Fine-tuned	33.7	12.4	33.9	79.2