SHAPED: Shared-Private Encoder-Decoder for Text Style Adaptation

Ye Zhang[*]
UT Austin
yezhang@utexas.edu

Nan Ding   **Radu Soricut**
Google AI
{dingnan,rsoricut}@google.com

Abstract

Supervised training of abstractive language generation models results in learning conditional probabilities over language sequences based on the supervised training signal. When the training signal contains a variety of writing styles, such models may end up learning an 'average' style that is directly influenced by the training data make-up and cannot be controlled by the needs of an application. We describe a family of model architectures capable of capturing both generic language characteristics via shared model parameters, as well as particular style characteristics via private model parameters. Such models are able to generate language according to a specific learned style, while still taking advantage of their power to model generic language phenomena. Furthermore, we describe an extension that uses a mixture of output distributions from all learned styles to perform on-the-fly style adaptation based on the textual input alone. Experimentally, we find that the proposed models consistently outperform models that encapsulate single-style or average-style language generation capabilities.

1   Introduction

Encoder-decoder models have recently pushed forward the state-of-the-art performance on a variety of language generation tasks, including machine translation (Bahdanau et al., 2015; Wu et al., 2016; Vaswani et al., 2017), text summarization (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017), dialog systems (Li et al., 2016; Asghar et al., 2017), and image captioning (Xu et al., 2015; Ranzato et al., 2015; Liu et al., 2017). This framework consists of an encoder that reads the input data and encodes it as a sequence of vectors, which is in turn used by a decoder to generate an-

other sequence of vectors used to produce output symbols step by step.

The prevalent approach to training such a model is to update all the model parameters using all the examples in the training data (over multiple epochs). This is a reasonable approach, under the assumption that we are modeling a single underlying distribution in the data. However, in many applications and for many natural language datasets, there exist multiple underlying distributions, characterizing a variety of language styles. For instance, the widely-used Gigaword dataset (Graff and Cieri, 2003) consists of a collection of articles written by various publishers (The New York Times, Agence France Presse, Xinhua News, etc.), each with its own style characteristics. Training a model's parameters on all the training examples results in an averaging effect across style characteristics, which may lower the quality of the outputs; additionally, this averaging effect may be completely undesirable for applications that require a level of control over the output style. At the opposite end of the spectrum, one can choose to train one independent model per each underlying distribution (assuming we have the appropriate signals for identifying them at training time). This approach misses the opportunity to exploit common properties shared by these distributions (e.g., generic characteristics of a language, such as noun-adjective position), and leads to models that are under-trained due to limited data availability per distribution.

In order to address these issues, we propose a novel neural architecture called SHAPED (shared-private encoder-decoder). This architecture has both shared encoder/decoder parameters that are updated based on all the training examples, as well as private encoder/decoder parameters that are updated using only examples from their corresponding underlying training distributions. In addition

---

[*] Work done as an intern at Google AI.

to learning different parametrization between the shared model and the private models, we jointly learn a classifier to estimate the probability of each example belonging to each of the underlying training distributions. In such a setting, the shared parameters ('shared model') are expected to learn characteristics shared by the entire set of training examples (i.e., language generic), whereas each private parameter set ('private model') learns particular characteristics (i.e., style specific) of their corresponding training distribution. At the same time, the classifier is expected to learn a probability distribution over the labels used to identify the underlying distributions present in the input data. At test time, there are two possible scenarios. In the first one, the input signal explicitly contains information about the underlying distribution (e.g., the publisher's identity). In this case, we feed the data into the shared model and also the corresponding private model, and perform sequence generation based on a concatenation of their vector outputs; we refer to this model as the SHAPED model. In a second scenario, the information about the underlying distribution is either not available, or it refers to a distribution that was not seen during training. In this case, we feed the data into the shared model and all the private models; the output distribution of the symbols of the decoding sequence is estimated using a mixture of distributions from all the decoders, weighted according to the classifier's estimates for that particular example; we refer to this model as the Mix-SHAPED model.

We test our models on the headline-generation task based on the aforementioned Gigaword dataset. When the publisher's identity is presented as part of the input, we show that the SHAPED model significantly surpasses the performance of the shared encoder-decoder baseline, as well as the performance of private models (where one individual, per-publisher model is trained for each in-domain style). When the publisher's identity is not presented as part of the input (i.e., not presented at run-time but revealed at evaluation-time for measurement purposes), we show that the Mix-SHAPED model exhibits a high level of classification accuracy based on textual inputs alone (accuracy percentage in the 80s overall, varying by individual publisher), while its generation accuracy still surpasses the performance of the baseline models. Finally, when the publisher's identity

is unknown to the model (i.e., a publisher that was not part of the training dataset), we show that the Mix-SHAPED model performance far surpasses the shared model performance, due to the ability of the Mix-SHAPED model to perform on-the-fly adaptation of output style. This feat comes from our model's ability to perform two distinct tasks: match the incoming, previously-unseen input style to existing styles learned at training time, and use the correlations learned at training time between input and output style characteristics to generate style-appropriate token sequences.

2   Related Work

Encoder-Decoder Models for Structured Output Prediction

Encoder-decoder architectures have been successfully applied to a variety of structure prediction tasks recently. Tasks for which such architectures have achieved state-of-the-art results include machine translation (Bahdanau et al., 2015; Wu et al., 2016; Vaswani et al., 2017), automatic text summarization (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Paulus et al., 2017; Nema et al., 2017), sentence simplification (Filippova et al., 2015; Zhang and Lapata, 2017), dialog systems (Li et al., 2016, 2017; Asghar et al., 2017), image captioning (Vinyals et al., 2015; Xu et al., 2015; Ranzato et al., 2015; Liu et al., 2017), etc. By far the most used implementation of such architectures is based on the original sequence-to-sequence model (Sutskever et al., 2014), augmented with its attention-based extension (Bahdanau et al., 2015). Although our SHAPED and Mix-SHAPED model formulations do not depend on a particular architecture implementation, we do make use of the (Bahdanau et al., 2015) model to instantiate our models.

Domain Adaptation for Neural Network Models

One general approach to domain adaptation for natural language tasks is to perform data/feature augmentation that represents inputs as both general and domain-dependent data, as originally proposed in (Daumé III, 2009), and ported to neural models in (Kim et al., 2016). For computer vision tasks, a line of work related to our approach has been proposed by Bousmalis et al. (2016) using what they call domain separation networks. As a tool for studying unsupervised domain adaptation

for image recognition tasks, their proposal uses CNNs for encoding an image into a feature representation, and also for reconstructing the input sample. It also makes use of a private encoder for each domain, and a shared encoder for both the source and the target domain. The approach we take in this paper shares this idea of model parametrization according to the domain/style, but goes further with the Mix-SHAPED model, performing on-the-fly adaptation of the model outputs. Other CNN-based domain adaptation methods for object recognition tasks are presented in (Long et al., 2016; Chopra et al., 2013; Tzeng et al., 2015; Sener et al., 2016).

For NLP tasks, Peng and Dredze (2017) take a multi-task approach to domain adaptation and sequence tagging. They use a shared encoder to represent instances from all of the domains, and use a domain projection layer to project the shared layer into a domain-specific space. They only consider the supervised domain-adaptation case, in which labeled training data exists for the target domain. Glorot et al. (2011) use auto-encoders for learning a high-level feature extraction across domains for sentiment analysis, while Zhou et al. (2016) employ auto-encoders to directly transfer the examples across different domains also for the same sentiment analysis task. Hua and Wang (2017) perform an experimental analysis on domain adaptation for neural abstractive summarization.

An important requirement of all the methods in the related work described above is that they require access to the (unlabeled) target domain data, in order to learn a domain-invariant representation across source and target domains. In contrast, our Mix-SHAPED model does not need access to a target domain or style at training time, and instead performs the adaptation on-the-fly, according to the specifics of the input data and the correlations learned at training time between available input and output style characteristics. As such, it is a more general approach, which allows adaptation for a much larger set of target styles, under the weaker assumption that there exists one or more styles present in the training data that can act as representative underlying distributions.

3 Model Architecture

Generally speaking, a standard encoder-decoder model has two components: an encoder that takes as input a sequence of symbols $\mathbf{x} =$

$(x_1, x_2, ..., x_{T_x})$ and encodes them into a set of vectors $\mathbf{H} = (h_1, h_2, ..., h_{T_x})$,

$$\mathbf{H} = f_{\text{enc}}(\mathbf{x}), \quad (1)$$

where f_{enc} is the computation unit in the encoder; and, a decoder that generates output symbols at each time stamp t, conditioned on $\mathbf{H}$ as well as the decoder inputs $\mathbf{y}_{1:t-1}$,

$$s_t = f_{\text{dec}}(\mathbf{y}_{1:t-1}, \mathbf{H}), \quad (2)$$

where f_{dec} is the computation unit in the decoder. Instantiations of this framework include the widely-used attention-based sequence-to-sequence model (Bahdanau et al., 2015), in which f_{enc} and f_{dec} are implemented by an RNN architecture using LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Chung et al., 2014) units. A more recent instantiation of this architecture is the Transformer model (Vaswani et al., 2017), built using self-attention layers.

3.1 SHAPED: Shared-private encoder-decoder

The abstract encoder-decoder model described above is usually trained over all examples in the training data. We call such a model a *shared* encoder-decoder model, because the model parameters are shared across all training and test instances. Formally, the shared encoder-decoder consists of the computation units f_{enc}^s and f_{dec}^s. Given an instance $\mathbf{x}$, it generates a sequence of vectors $\mathbf{S}^s = (s_1^s, ... s_T^s)$ by:

$$\mathbf{H}^s = f_{\text{enc}}^s(\mathbf{x}), s_t^s = f_{\text{dec}}^s(\mathbf{y}_{1:t-1}, \mathbf{H}^s). \quad (3)$$

The drawback of the shared encoder-decoder is that it fails to account for particular properties of each style that may be present in the data. In order to capture such particular style characteristics, a straightforward solution is to train a *private* model for each style. Assuming a style set $\mathbb{D} = \{D_1, D_2..., D_{|\mathbb{D}|}\}$, such a solution implies that each style has its own private encoder computation unit and decoder computation unit. At both training and testing time, each private encoder and decoder only process instances that belong to their own style. Given an instance along with its style $(\mathbf{x}, z)$ where $z \in \{1, \ldots, |\mathbb{D}|\}$, the private encoder-decoder generates a sequence of vectors $\mathbf{S}^z = (s_1^z, ... s_T^z)$ by:

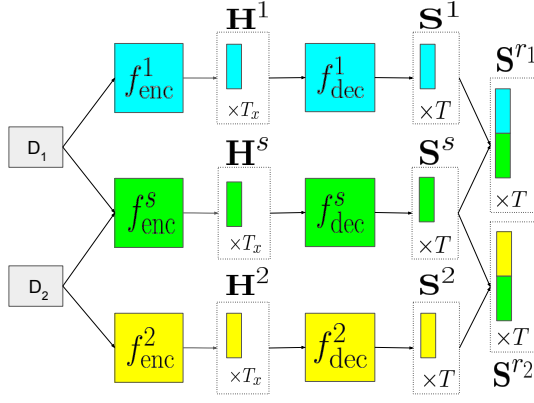$$\mathbf{H}^z = f_{\text{enc}}^z(\mathbf{x}), s_t^z = f_{\text{dec}}^z(\mathbf{y}_{1:t-1}, \mathbf{H}^z). \quad (4)$$

Figure 1: Illustration of the SHAPED model using two styles D_1 and D_2. D_1 articles pass through the private encoder f^1_{enc} and decoder f^1_{dec}. D_2 articles pass through the private encoder f^2_{enc} and decoder f^2_{dec}. Both of them also go through the shared encoder f^s_{dec} and decoder f^s_{dec}.

Although the private encoder/decoder models do preserve style characteristics, they fail to take into account the common language features shared across styles. Furthermore, since each style is represented by a subset of the entire training set, such private models may end up as under-trained, due to limited number of available data examples.

In order to efficiently capture both common and unique features of data with different styles, we propose the SHAPED model. In the SHAPED model, each data-point goes through both the shared encoder-decoder and its corresponding private encoder-decoder. At each step of the decoder, the output from private and shared ones are concatenated to form a new vector:

$$s^{r_z}_t = [s^z_t, s^s_t], \qquad (5)$$

that contains both private features for style z and shared features induced from other styles, as illustrated in Fig 1. The output symbol distribution over tokens $o_t \in V$ (where V is the output vocabulary) at step t is given by:

$$p(o_t|\mathbf{x}, y_{1:t-1}, z) = \text{Softmax}(g(s^{r_z}_t)), \quad (6)$$

where g is a multi-layer feed-forward network that maps $s^{r_z}_t$ to a vector of size $|V|$. Given N training examples $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, z^{(1)}), \ldots, (\mathbf{x}^{(N)}, \mathbf{y}^{(N)}, z^{(N)})$, the conditional probability of the output $\mathbf{y}^{(i)}$ given

article $\mathbf{x}^{(i)}$ and its style $z^{(i)} \in \{1, \ldots, |\mathbb{D}|\}$ is:

$$p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, z^{(i)}) = \prod_t p(o_t = y^{(i)}_t|\mathbf{x}^{(i)}, \mathbf{y}^{(i)}_{1:t-1}, z^{(i)}).$$
$$(7)$$

At inference time, given an article $\mathbf{x}$ with style z, we feed $\mathbf{x}$ into $f^s_{\text{enc}}, f^s_{\text{dec}}, f^z_{\text{enc}}, f^z_{\text{dec}}$ (Eq. 3-4) and obtain symbol distributions at each step t using Eq. 6. We sample from the distribution and obtain a symbol o_t which will be used as the estimated y_t and fed to the next steps.

3.2 The Mix-SHAPED Model

One limitation of the above model is that it can only handle test data containing an explicit style label from $\mathbb{D} = \{D_1, D_2..., D_{|\mathbb{D}|}\}$. However, there is frequently the case that, at test time, the style label is not present as part of the input, or that the input style is not part of the modeled set $\mathbb{D}$.

We treat both of these cases similarly, as a case of modeling an unknown style. We first describe our treatment of such a case at run-time. We use a latent random variable $z \in \{1, \ldots, |\mathbb{D}|\}$ to denote the underlying style of a given input. When generating a token at step t, the output token distribution takes the form of a mixture of SHAPED (Mix-SHAPED) model outputs:

$$p(o_t|\mathbf{x}, \mathbf{y}_{1:t-1}) =$$
$$\sum_{d=1}^{|\mathbb{D}|} p(o_t|\mathbf{x}, \mathbf{y}_{1:t-1}, z = d)p(z = d|\mathbf{x}), \qquad (8)$$

where $p(o_t|\mathbf{x}, \mathbf{y}_{1:t-1}, z = d)$ is the output symbol distribution of SHAPED decoder d, evaluated as in Eq. 6. Fig. 2 contains an illustration of such a model. In this formulation, $p(z|\mathbf{x})$ denotes the style conditional probability distribution from a trainable style classifier.

The joint data likelihood of target sequence $\mathbf{y}$ and target domain label z for input sequence $\mathbf{x}$ is:

$$p(\mathbf{y}, z|\mathbf{x}) = p(\mathbf{y}|z, \mathbf{x}) \cdot p(z|\mathbf{x}) \qquad (9)$$

Training the Mix-SHAPED model involves minimizing a loss function that combines the negative log-likelihood of the style labels and the negative log-likelihood of the symbol sequences (see the model in Fig 3):

$$\text{Loss}_{\text{Mix-SHAPED}} = -\sum_{i=1}^N \log p(z^{(i)}|\mathbf{x}^{(i)})$$
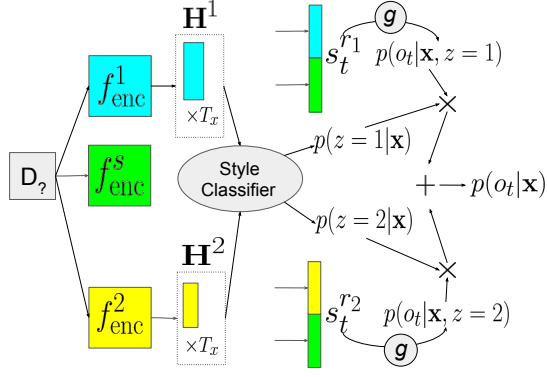$$-\sum_{i=1}^N \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, z^{(i)}). \qquad (10)$$

Figure 2: Decoding data with unknown style using a Mix-SHAPED model. The data is run through all encoders and decoders. The output of private encoders is fed into a classifier that estimates style distribution. The output symbol distribution is a mixture over all decoder outputs.

At run-time, if the style d of the input is available and $d \in \mathbb{D}$, we decode the sequence using Eq. 6. This also corresponds to the case $p(z = d|\mathbf{x}) = 1$ and 0 for all other styles, and reduces Eq. 8 to Eq. 6. If the style of the input is unknown (or known, but with $d' \notin \mathbb{D}$), we decode the sequence using Eq. 8, in which case the mixture over SHAPED models given by $p(z|\mathbf{x})$ is approximating the desired output style.

4 Model Instantiation

As an implementation of the encoder-decoder model, we use the attention-based sequence-to-sequence model from (Bahdanau et al., 2015), with an RNN architecture using GRU units (Chung et al., 2014). The input token sequences are first projected into an embedding space via an embedding matrix $\mathbf{E}$, resulting in a sequence of vectors as input representations.

The private and shared RNN cells generate a sequence of hidden state vectors $\mathbf{H}^z = \{h_j^z\}$, $z \in \{1, ..., |\mathbb{D}|\}$ and $\mathbf{H}^s = \{h_j^s\}$, for $j \in \{1, ..., T_x\}$. At each step in the encoder, h_j^z and h_j^s are concatenated to form a new output vector $h_j^{rz} = [h_j^z, h_j^s]$. The final state of each encoder is used as the initial state of the corresponding decoder. At time step t in the decoder, the private and shared RNN cell first generate hidden state vectors $\{s_t^z\}$, $z \in \{1, ..., |\mathbb{D}|\}$ and s_t^s, then s_t^s is concatenated with each s_t^z to form new vectors $\{s_t^{rz}\}$ ($z \in \{1, ..., |\mathbb{D}|\}$).

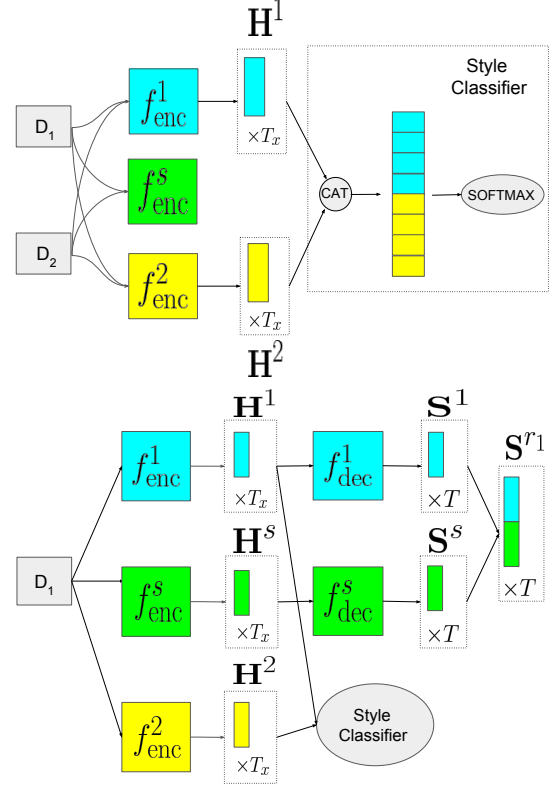We apply the attention mechanism on s_t^{rz}, using

Figure 3: Training a Mix-SHAPED model. (a) Each example is fed to all private encoders f_{enc}^1, f_{enc}^2, whose outputs are concatenated and fed to a style classifier. (b) The D_1 examples only use $f_{enc}^1, f_{dec}^1, f_{enc}^s, f_{dec}^s$ to decode texts. Private encoder-decoders of other styles are not used.

attention weights calculated as:

$$q_{tj}^{rz} = v_a \tanh(W_a h_j^{rz} + U_a s_t^{rz}), \quad (11)$$

which are normalized to a probability distribution:

$$\alpha_{tj}^{rz} = \frac{\exp(q_{tj}^{rz})}{\sum_{i=1}^{T_x} \exp(q_{ti}^{rz})} \quad (12)$$

Context vectors are computed using normalized attention weights:

$$c_t^{rz} = \sum_{j=1}^{T_x} \alpha_{tj}^{rz} h_j^{rz} \quad (13)$$

Given the context vector and the hidden state vectors, the symbol distribution at step t is:

$$p(o_t|\mathbf{x}, \mathbf{y}_{1:t}, z) = \text{softmax}(g([c_t^{rz}, s_t^{rz}])) \quad (14)$$

The attention weights in W_a, U_a, and v_a, as well as the embedding matrix $\mathbf{E}$ and vocabulary V are shared by all encoders and decoders. We use Eq. 14 to calculate the symbol loss in Eq. 10.

5 Quantitative Experiments

We perform a battery of quantitative experiments, designed to answer several main questions: 1) Do the proposed model improve generation performance over alternative approaches? 2) Can a style classifier built using an auxiliary loss provide a reliable estimate on text style? 3) In the case of unknown style, does the Mix-SHAPED model improve generation performance over alternative approaches? 4) To what extent do our models capture style characteristics as opposed to, say, content characteristics?

We perform our experiments using text summarization as the main task. More precisely, we train and evaluate headline generation models using the publicly-available Gigaword dataset (Graff and Cieri, 2003; Napoles et al., 2012).

5.1 Headline-generation Setup

The Gigaword dataset contains news articles from seven publishers: Agence France-Presse (AFP), Associated Press Worldstream (APW), Central News Agency of Taiwan (CNA), Los Angeles Times/Washington Post Newswire Service (LTW), New York Times (NYT), Xinhua News Agency (XIN), and Washington Post/Bloomberg Newswire Service (WPB). We pre-process this dataset in the same way as in (Rush et al., 2015), which results in articles with average length 31.4 words, and headlines with average length 8.5 words.

We consider the publisher identity as a proxy for style, and choose to model as in-domain styles the set $\mathbb{D} = \{$AFP, APW, NYT, XIN$\}$, while holding out CNA and LTW for out-of-domain style testing. This results in a training set containing the following number of (article, headline) instances: 993,584 AFP, 1,493,758 APW, 578,259 NYT, and 946,322 XIN. For the test set, we sample a total number of 10,000 in-domain examples from the original Gigawords test dataset, which include 2,886 AFP, 2,832 APW, 1,610 NYT, and 2,012 XIN. For out-of-domain testing, we randomly sample 10,000 LTW and 10,000 CNA test data examples. We remove the WPB articles due to their small number of instances.

5.1.1 Experimental Setup

We compare the following models:

- A Shared encoder-decoder model (S) trained on all styles in $\mathbb{D}$;

- A suite of Private encoder-decoder models (P), each one trained on a particular style from $\mathbb{D} = \{$AFP, APW, NYT, XIN$\}$;[1]

- A SHAPED model (SP) trained on all styles in $\mathbb{D}$; at test time, the style of test data is provided to the model; the article is only run through its style-specific private network and shared network (style classifier is not needed);

- A Mix-SHAPED model (M-SP) trained on all styles in $\mathbb{D}$; at test time, the style of article is not provided to the model; the output is computed using the mixture model, with the estimated style probabilities from the style classifier used as weights.

When testing on the out-of-domain styles CNA/LTW, we only compare the Shared (S) model with the Mix-SHAPED (M-SP) model, as the others cannot properly handle this scenario.

As hyper-parameters for the model instantiation, we used 500-dimension word embeddings, and a three-layer, 500-dimension GRU-cell RNN architecture; the encoder was instantiated as a bidirectional RNN. The lengths of the input and output sequences were truncated to 40 and 20 tokens, respectively. All the models were optimized using Adagrad (Duchi et al., 2011), with an initial learning rate of 0.01. The training procedure was done over mini-batches of size 128, and the updates were done asynchronously across 40 workers for 5M steps. The encoder/decoder word embedding and the output projection matrices were tied to minimize the number of parameters. To avoid the slowness from the softmax operator over large vocabulary sizes, and also mitigate the impact of out-of-vocabulary tokens, we applied a subtokenization method (Wu et al., 2016), which invertibly transforms a native token into a sequence of subtokens from a limited vocabulary (here set to 32K).

Comparison with Previous Work In the next section, we report our main results using the in-domain and out-of-domain (w.r.t. the selected publisher styles) test sets described above, since these test sets have a balanced publisher style frequency that allows us to measure the impact of our style-adaptation models. However, we also report

---

[1] We also tried to warm-start a private model using the best checkpoint of the shared model, but found that it cannot improve over the shared model.

|	AFP/APW/XIN/NYT Test		
	Rouge-1	Rouge-2	Rouge-L
P	39.14±0.47	19.74±0.48	36.42±0.46
S	39.32±0.26	19.63±0.24	36.51±0.26
SP	**40.34**±0.26	**20.38**±0.25	**37.52**±0.25
M-SP	40.10±0.25	20.21±0.26	37.30±0.26

Table 1: ROUGE F1 scores on the combined AFP/APW/XIN/NYT in-domain test set.

here the performance of our Shared (S) baseline model (with the above hyper-parameters) on the original 2K test set used in (Rush et al., 2015). On that test set, our S model obtains 30.13 F1 ROUGE-L score, compared to 28.34 ROUGE-L obtained by the ABS+ model (Rush et al., 2015), and 30.64 ROUGE-L obtained by the words-lvt2k-1sent model (Nallapati et al., 2016). This comparison indicates that our S model is a competitive baseline, making the comparisons against the SP and M-SP models meaningful when using our in-domain and out-of-domain test sets.

5.1.2 Main Results

The Rouge scores for the in-domain testing data are reported in Table 1 (over the combined AFP/APW/XIN/NYT testset) and Fig. 4a (over individual-style test sets). The numbers indicate that the SP and M-SP models consistently outperform the S and P model, supporting the conclusion that the S model loses important characteristics due to averaging effects, while the P models miss the opportunity to efficiently exploit the training data. Additionally, the performance of SP is consistently better than M-SP in this setting, which indicates that the style label is helpful. As shown in Fig. 4b, the style classifier achieves around 80% accuracy overall in predicting the style under the M-SP model, with some styles (e.g., XIN) being easier to predict than others. The performance of the classifier is directly reflected in the quantitative difference between the SP and M-SP models on individual-style test sets (see Fig. 4a, where the XIN style has the smallest difference between the two models).

The evaluation results for the out-of-domain scenario are reported in Table 2. The numbers indicate that the M-SP model significantly outperforms the S model, supporting the conclusion that the M-SP model is capable of performing on-the-fly adaptation of output style. This conclusion is further strengthened by the style probability distributions shown in Fig 5: they indicate that, for
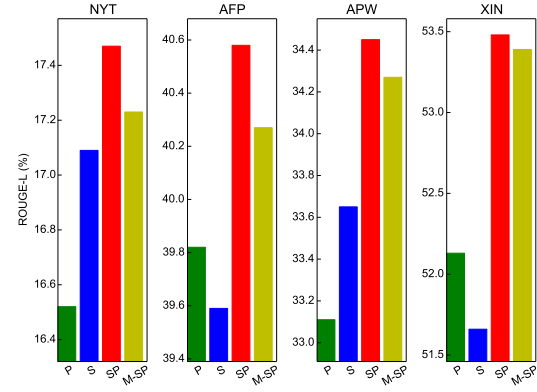
(a) Rouge-L scores on headline generation, shown separately on four in-domain styles.
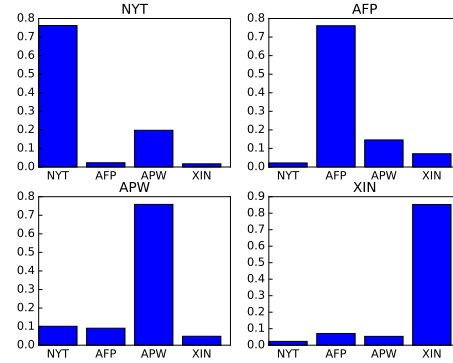
(b) Average estimated probability distribution by the M-SP model over the four styles, for each in-domain target style in the test set.

Figure 4: Experimental results on the headline generation task, for in-domain styles.

the out-of-domain CNA style, the output mixture is heavily weighted towards the XIN style (0.6 of the probability mass), while for the LTW style, the output mixture weights heavily the NYT style (0.72 of the probability mass). This result is likely to reflect true style characteristics shared by these publishers, since both CNA and XIN are produced by Chinese news agencies (from Taiwan and mainland China, respectively), while both LTW and NYT are U.S. news agencies owned by the same media corporation.

5.1.3 Experiment Variants

Model capacity In order to remove the possibility that the improved performance of the SP model is due simply to an increased model size compared to the S model, we perform an experiment in which we triple the size of the GRU cell dimensions for the S model. However, we find no significant performance difference compared to the

	CNA Test			LTW Test		
	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
S	$40.73_{\pm0.21}$	$17.75_{\pm0.18}$	$37.70_{\pm0.20}$	$27.08_{\pm0.19}$	$8.97_{\pm0.15}$	$25.01_{\pm0.17}$
M-SP	$\mathbf{42.00}_{\pm0.20}$	$\mathbf{19.48}_{\pm0.21}$	$\mathbf{39.24}_{\pm0.22}$	$\mathbf{27.79}_{\pm0.19}$	$\mathbf{9.31}_{\pm0.18}$	$\mathbf{25.60}_{\pm0.17}$

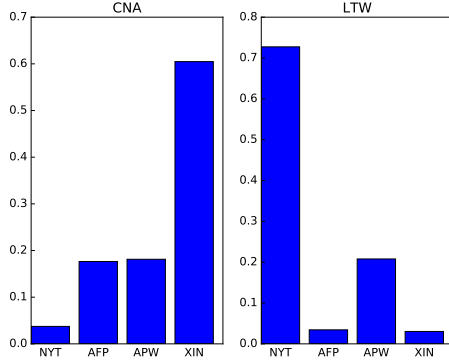Table 2: ROUGE F1 scores on out-of-domain style test sets CNA and LTW.

Figure 5: Estimated style probabilities over the four in-domain styles AFP/APW/XIN/NYT, for out-of-domain styles CNA and LTW.

original dimensions (the ROUGE-L score of the triple-size S model is 36.61, compared to 36.51 obtained of the original S model).

Style embedding A competitive approach to modeling different styles is to directly encode the style information into the embedding space. In (Johnson et al., 2016), the style label is converted into a one-hot vector and is concatenated with the word embedding at each time step in the S model. The outputs of this model are at 36.68 ROUGE-L, slightly higher than the baseline S model, but significantly lower than the SP model performance (37.52 ROUGE-L).

Another style embedding approach is to augment the S model with continuous trainable style embeddings for each predefined style label, similar to (Ammar et al., 2016). The resulting outputs achieve 37.2 ROUGE-L, which is better than the S model with one-hot style embedding, but still worse than the SP method (statistically significant at p-value=0.025 using paired t-test). However, neither of these approaches apply to the cases when the style is out-of-domain or unknown during testing. In contrast, such cases are handled naturally by the proposed M-SP model.

Ensemble model Another question is whether the SP model simply benefits from ensembling

multiple models rather than style adaptation. To answer this question, we apply a uniform mixture over the private model output along with the shared model output, rather than using the learnt probability distribution from the style classifier. The ROUGE-1/2/L scores are 39.9/19.7/37.0. They are higher than the S model but significantly lower than the SP model and the M-SP model (p-value 0.016). This result confirms that the information that the style classifier encodes is beneficiary, and leads to improved performance.

Style vs. Content Previous experiments indicate that the SP and M-SP models have superior generation accuracy, but it is unclear to what extent the difference comes from improved modeling of style versus modeling of content. To clarify this issue, we performed an experiment in which we replace the named entities appearing in both article and headline with corresponding entity tags, in effect suppressing almost completely any content signal. For instance, given an input such as "China called Thursday on the parties involved in talks on North Korea's nuclear program to show flexibility as a deadline for implementing the first steps of a breakthrough deal approached.", paired with goldtruth output "China urges flexibility as NKorea deadline approaches", we replaced the named entities with their types, and obtained: "LOC_0 called Thursday on the ORG_0 involved in NON_2 on LOC_1 's NON_3 to show NON_0 as a NON_1 for implementing the first NON_4 of a NON_5 approached .", paired with "LOC_0 urges NON_0 as LOC_1 NON_1 approaches."

Under this experimental conditions, both the SP and M-SP models still achieve significantly better performance compared to the S baseline. On the combined AFP/APW/XIN/NYT in-domain test set, the SP model achieves 61.70 ROUGE-L and M-SP achieves 61.52 ROUGE-L, compared to 60.20 ROUGE-L obtained by the S model. On the CNA/LTW out-of-domain test set, M-SP achieves 60.75 ROUGE-L, compared to 59.47 ROUGE-L by the S model.

In Table 3, we show an example which indi-

article	the org_2 is to forge non_1 with the org_3 located in loc_2 , loc_1 , the per_0 of the loc_0 org_4 said tuesday .
title	loc_0 org_0 to forge non_0 with loc_1 org_1
output by S	org_0 to org_1 in non_0
output by M-SP	loc_0 org_0 to forge non_0 with loc_1 org_1
article	loc_0 - born per_0 per_0 will pay non_1 here next month to per_1 , the org_2 (org_1) per_1 who per_1 perished in an non_2 in february , the org_3 said thursday .
title	per_0 to pay non_0 to late org_1 org_0
output by S	per_0 to visit org_0 in non_0
output by M-SP	per_0 to pay non_0 to org_1 org_0

Table 3: Examples of input article (and groundtruth title) and output generated by S and M-SP. Named entities in the training instances (both article and title) are replaced the entity type.

cates the ability of style adaptation benifiting summarization. For instance, we find that both CNA and XIN make more frequent use of the style pattern "xxx will/to [verb] yyy . . ., zzz said ???day" (about 15% of CNA articles contain this pattern, while only 2% of LTW articles have it). From Table 3, we can see that the S model sometimes misses or misuses the verb in its output, while the M-SP model does a much better job at capturing both the verb/action as well as other relations (via prepositions, etc.)

Fig. 6 shows the estimated style probabilities over the four styles AFP/APW/XIN/NYT for CNA and LTW, under this experiment condition. We observe that, in this version as well, CNA is closely matching the style of XIN, while LTW is matching that of NYT. The distribution is similar to the one in Fig. 5, albeit a bit flatter as a result of content removal. As such, it supports the conclusion that the classifier indeed learns style (in addition to content) characteristics.

6 Conclusion

In this paper, we describe two new style-adaptation model architectures for text sequence generation tasks, SHAPED and Mix-SHAPED. Both versions are shown to significantly outperform models that are either trained in a manner that ignores style characteristics (and hence exhibit a style-averaging effect in their outputs), or models that are trained single-style.

The latter is a particularly interesting result, as a model that is trained (with enough data) on a single-style and evaluated on the same style would be expected to exhibit the highest performance. Our results show that, even for single-style models
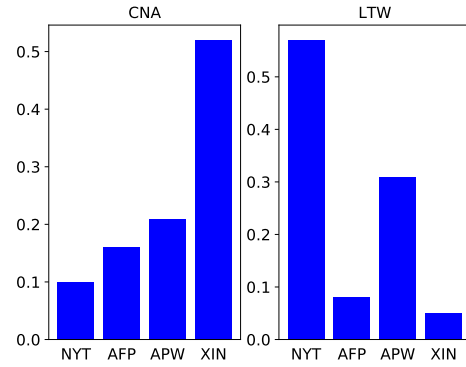

Figure 6: Estimated style probabilities over the four in-domain styles AFP/APW/XIN/NYT, for out-of-domain styles CNA and LTW, after named entities in the article and summary are replaced with entity tags.

trained on over 1M examples, their performance is inferior to the performance of SHAPED models on that particular style.

Our conclusion is that the proposed architectures are both efficient and effective in modeling both generic language phenomena, as well as particular style characteristics, and are capable of producing higher-quality abstractive outputs that take into account style characteristics.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *TACL* 4:431–444.

Nabiha Asghar, Pascal Poupart, Xin Jiang, and Hang Li. 2017. Deep active learning for dialogue generation. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*. pages 78–83.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*. pages 343–351.

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *HLT-NAACL*. pages 93–98.

Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. 2013. Dlid: Deep learning for domain adaptation by interpolating between domains. In *ICML workshop on challenges in representation learning*. volume 2.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815* .

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. pages 513–520.

David Graff and C Cieri. 2003. English gigaword corpus. *Linguistic Data Consortium* .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Xinyu Hua and Lu Wang. 2017. A pilot study of domain adaptation effect for neural abstractive summarization. *arXiv preprint arXiv:1707.07062* .

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR* abs/1611.04558. http://arxiv.org/abs/1611.04558.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541* .

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547* .

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. 2017. Optimization of image description metrics using policy gradient methods. In *International Conference on Computer Vision (ICCV)*.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2016. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*. pages 136–144.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of CoNLL*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, pages 95–100.

Preksha Nema, Mitesh Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. *arXiv preprint arXiv:1704.08300* .

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* .

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. *ACL 2017* page 91.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR* abs/1511.06732.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*. pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of ACL*.

Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. 2016. Learning transferrable representations for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*. pages 2110–2118.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. 2015. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 4068–4076.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 3156–3164.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. of the 32nd International Conference on Machine Learning (ICML)*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931* .

Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang, and Tingting He. 2016. Bi-transferring deep neural networks for domain adaptation. In *ACL (1)*.