Deep Transfer Reinforcement Learning for Text Summarization

Yaser Keneshloo
Discovery Analytics Center
Virginia Tech
yaserkl@vt.edu

Naren Ramakrishnan
Discovery Analytics Center
Virginia Tech
naren@vt.edu

Chandan K. Reddy
Discovery Analytics Center
Virginia Tech
reddy@cs.vt.edu

Abstract

Deep neural networks are data hungry models and thus face difficulties when attempting to train on small text datasets. Transfer learning is a potential solution but their effectiveness in the text domain is not as explored as in areas such as image analysis. In this paper, we study the problem of transfer learning for text summarization and discuss why existing state-of-the-art models fail to generalize well on other (unseen) datasets. We propose a reinforcement learning framework based on a self-critic policy gradient approach which achieves good generalization and state-of-the-art results on a variety of datasets. Through an extensive set of experiments, we also show the ability of our proposed framework to fine-tune the text summarization model using only a few training samples. To the best of our knowledge, this is the first work that studies transfer learning in text summarization and provides a generic solution that works well on unseen data.

Keywords: Transfer learning, text summarization, self-critic reinforcement learning.

1 Introduction

Text summarization is the process of summarizing a long document into few sentences that capture the essence of the whole document. In recent years, researchers have used news article datasets such as CNN/DM [10] and Newsroom [9] as a main resource for building and evaluating text summarization models. However, all these models suffer from a critical problem: *a model trained on a specific dataset works well only on that dataset.* For instance, if a model is trained on the CNN/DM dataset and tested on Newsroom dataset, the result is much poorer than when it is trained directly on the Newsroom dataset. This lack of generalization ability for current state-of-the-art models is the main motivation for our work.

This problem arises in situations where there is a need to perform summarization on a specific dataset, but either no ground-truth summaries exist for this dataset or collecting ground-truth summaries could be expensive and time-consuming. Thus, the only recourse in such a situation would be to simply apply a pre-trained summarization model to generate summaries for this data. However, as discussed in this paper, this approach will fail to satisfy the basic requirements of this task and thus fails to generate high quality summaries. Throughout our analysis, we work with two of the well-known news-related datasets for text summarization and one could expect that a model trained on either one of the datasets should perform well on the other or any news-related dataset. On the contrary, as shown in Table 1, the Fast-RL model [4] trained on CNN/DM, which holds the state-of-the-art result for text summarization task on CNN/DM test dataset with 41.18% F-score according to ROUGE-1 measure, will reach only 21.93% on this metric on Newsroom test data, a performance fall of almost 20%. This observation shows that these models suffer poor generalization capability.

In this paper, we first study the extent to which the current state-of-the-art models are vulnerable in generalizing to other datasets and discuss how transfer learning could help in alleviating some of these problems. In addition, we propose a solution based on reinforcement learning to remedy the generalization problem in text summarization model which achieves a good generalization score on a variety of summarization datasets. Traditional transfer learning usually works by pre-training a model using a large dataset and fine-tuning it on a target dataset and test the result on that target dataset. However, our proposed method, as shown in Fig 1, is able to achieve good results on a variety of datasets by only fine-tuning the model on a single dataset. Therefore, it removes the requirement of training separate transfer models for each dataset. To the best of our knowledge, this is the first work that studies transfer learning for the problem of text summarization and provides a solution for rectifying the generalization issue that arises in the current state-of-the-art summarization models. In addition, we conduct various experiments to demonstrate the ability of our proposed method to obtain state-of-the-art results on datasets with small amounts of ground-truth data.

Table 1: The Pointer-Generator [28] and Fast-RL [4] models are trained using CNN/DM dataset and tested on CNN/DM and Newsroom dataset.

	Pointer Generator [28]			**Fast-RL** [4]		
ROUGE	1	2	L	1	2	L
CNN/DM	39.77	17.15	36.18	41.18	18.19	38.78
Newsroom	26.59	14.09	23.44	21.93	9.37	19.61

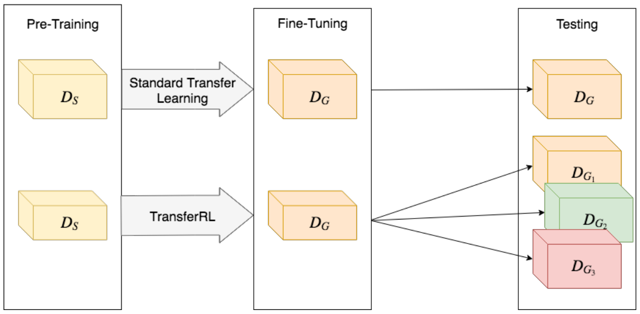


Figure 1: In standard transfer learning settings, a model is pre-trained on D_S and all the network layers are transferred and the model is fine-tuned using D_G and finally tested only on D_G. On the contrary, our proposed method (TransferRL) uses D_G to create a model that works good on a variety of datasets during test.

The rest of the paper is organized as follows: Section 2 describes transfer learning methods and the recent research works related to this problem, and Section 3 presents our proposed model for transfer learning in text summarization. Section 4 shows our experimental results and compare them with various benchmark datasets and Section 5 concludes our discussion.

2 Related Work

Recently, there has been a surge in the development of deep learning based methods for building models that have the ability to transfer and generalize to other similar problems. Transfer learning (TL) has been well-studied in the domain of image processing, however the effect of it on NLP problems is yet to be thoroughly investigated. In this section, we will review some of these works.

2.1 Transferring Trained Models In these works, the underlying model is first trained on a specific dataset and then used as a pre-trained model for another problem or dataset. In this method, depending on the underlying model, one can transform different types of neural network layers from the pre-trained model to the transfer model. Examples of these transferable layers are the word embedding layer, the convolutional layers in a CNN model, Fully Connected (FC) hidden layers, and finally the output layer [23]. Yosinski *et al.* [34] studied the effect of transferring different layers of a deep neural network and found that lower-level layers learn general features while higher level layers capture mostly the specific characteristic of the problem at hand. Specifically, researchers showed how one can transfer both low-level and high-level neural layers from a CNN for TL [5].

Recently, Semwal *et al.* [29] used this idea of transferring various network layers for text classification. Aside from transferring the network layers, they also experimented with freezing or fine-tuning these layers after the transfer and concluded that fine-tuning the transfer layers will always provide a better result. Moreover, TL has also been studied in the context of named entity recognition problem which is similar to a classification task [26, 14]. Our proposed method falls into this category. We not only study the effect of transferring network layers, but also propose a new co-training model for training text summarization model using reinforcement learning techniques.

2.2 Knowledge Distillation Knowledge distillation is a class of techniques that train a small network by transferring knowledge from a larger network. These techniques are typically used when we require building models for devices with limited computational power [1]. Usually, in these models, there is a teacher (larger model) and a student (smaller model) and the goal is to transfer knowledge from teacher to student. Recently, researchers have also used this idea to create models using meta-learning [27], few-shot learning [24, 31], one-shot learning [6, 3], and domain adaptation [7, 32], mostly for image classification problems. However, the effect of these types of models on NLP tasks is yet to be studied.

2.3 Building Generalized Models Recently, McCann *et al.* [16] released a challenge called Decathlon NLP which aims at solving ten different NLP problems with a single unified model. The main intuition behind this model is to comprehend the impact of transferring knowledge from different NLP tasks on building a generalized model that works well on every task. Although this model outperforms some of the state-of-the-art models in specific tasks, it fails to even reach baseline results in tasks like text summarization. We also observe such poor results from other generalized frameworks such as Google's Tensor2Tensor framework [33].

2.4 Text Summarization There is a vast amount of research work on the topic of text summarization using deep neural networks [30]. These works range from fully extractive methods [4, 19, 35] to completely abstractive ones [28, 12, 8]. As one of the earliest works on using neural networks for extractive summarization, Nallapati *et al.* [17] proposed a framework that used a ranking technique to extract the most salient sentence in the input. On the other hand, for abstractive summarization, it was Rush *et al.* [25] that for the first time used attention over a sequence-to-sequence (seq2seq) model for the problem of headline generation. To further improve the performance of these models, pointer-generator model [18] was proposed for successfully handling the Out-of-Vocabulary (OOV) words. This model was later improved by using the coverage mechanism [28]. However, all these models, suffer from a common problem known as *exposure bias* which refers to the fact that, during training, the model is trained by feeding ground-truth input at each decoder step, while during the test, it should rely on its own output to generate the next token. Also, the training is typically done using cross-entropy loss, while during test, metrics such as ROUGE [15] or BLEU [20] are used to evaluate the model. To tackle this problem, researchers suggested various models using scheduled sampling [2] and reinforcement learning based models [21, 4].

Recently, several authors have investigated methods which try to first perform extractive summarization by selecting the most salient sentences within a document using a classifier and then apply a language model or a paraphrasing model [13] on these selected sentences to get the final abstractive summarization [4, 19, 35]. However, none of these models, as discussed in this paper (and shown in Table 1) have the capability to generalize to other datasets and they only perform well for a specific dataset which was used as target data during the pre-training process.

3 Proposed Model

In this paper, we propose various transfer learning methods for the problem of text summarization. For all experiments, we consider two different datasets: *source dataset*, D_S, is the dataset which is used to train the pre-trained model, while *target dataset*, D_G[1], is the dataset that is used to fine-tune our pre-trained model. In light of transferring layers of a pre-trained models, our first proposed model transfers different layers of a pre-trained model trained using D_S and fine-tune using D_G. We then propose another method which uses a novel Reinforcement Learning (RL) framework to train the transfer model using training signals that is received from both D_S and D_G.

3.1 Transferring Network Layers There are various network layers that are used in a deep neural network. For instance, if the model has a CNN encoder and a LSTM decoder, the CNN layers and the hidden decoder layers trained on D_S could be used to fine-tune using D_G. Moreover, the word embedding is another important layer in this model. Either, we use pre-trained word embeddings such as Glove [22] during the training of D_S or let D_S train its own word embedding. We can still let the model to fine-tune a pre-trained word embedding during the training of the model. In summary, we can transfer the embedding layer, convolutional layer (if using CNN), hidden layers (if using LSTM), and the output layer in a text summarization transfer learning problem. One way to understand the effect of each of these layers is to fine-tune or freeze these layers during model transfer and report the best performing model. However, as suggested by [29], the best performing model occurs when all layers of a pre-trained model on D_S are transferred and let the model to fine-tune itself using D_G. Therefore, we follow the same practice and let the transferred model to fine-tune all trainable variables in our model. As shown later in the experimental result section, this way of transferring network layers provide a strong baseline in text summarization and the performance of our proposed reinforced model is close to this baseline. However, one of the main problems with this approach is that, not only the source dataset D_S should contain a large number of training samples but also D_G must have a lot of training samples to be able to fine-tune the pre-trained model and generalize the distribution of the pre-trained model parameters. Therefore, a successful transfer learning using this method requires a large number of samples both for D_S and D_G. This could be problematic, specifically for cases where the target dataset is small and fine-tuning a model will cause overfitting. For these reasons, we will propose a model which uses reinforcement learning to fine-tune the model only based on the reward that is obtained from the target dataset.

3.2 Transfer Reinforcement Learning (TransferRL) In this section, we explain our proposed reinforcement learning based framework for knowledge transfer in text summarization. The basic underlying summarization mechanism used in our work is the pointer-generator model [28].

3.2.1 Why Pointer-Generator? The reason we choose a pointer-generator model as the basis of our framework is its ability in handling Out-of-Vocabulary (OOV) words which is necessary for transfer learning. Note that once a specific vocabulary generated from D_S is used to train the pre-trained model, we cannot use a different set of vocabulary during fine-tuning stage on D_G, since the indexing of words could change for words

[1] Note that, we use D_G instead of D_T for the target dataset to avoid confusing this T subscript with time.

in the second dataset[2]. According to our experiments, amongst the top 50K words in CNN/DM and Newsroom datasets, only 39K words are common between the two datasets and thus a model trained on each of these datasets will have more than 11K OOVs during the fine-tuning step. Therefore, a framework that is not able to handle these OOV words will have significantly poor results after the transfer. One naïve approach in resolving this problem could be to use a shared set of vocabulary between D_S and D_G. However, such a model will still suffer from the generalization to other datasets with a different set of vocabulary.

3.2.2 Pointer-Generator As shown in Fig 2, a pointer-generator model comprises of a series of LSTM encoders (blue boxes) and LSTM decoders (green boxes). Let us consider dataset $D = \{d_1, \cdots, d_N\}$ as a dataset that contains N documents along with their summaries. Each document is represented by a series of T_e words, i.e. $d_i = \{x_1, \cdots, x_{T_e}\}$, where $x_t \in V = \{1, \cdots, |V|\}$. Each encoder takes the embedding of word x_t as the input and generates the output state h_t. The decoder, on the other hand, takes the last state from the encoder, i.e., h_{T_e}, and starts generating an output of size $T < T_e$, $\hat{\mathbf{Y}} = \{\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_T\}$, based on the current state of the decoder s_t, and the ground-truth summary word y_t. At each step of decoding j, the attention vector α_j, context vector c_j, and output distribution p_{vocab} can be calculated as follows:

(3.1)
$$\begin{aligned} f_{ij} &= v_1^T tanh(W_h h_i + W_s s_j + b_1) \\ \alpha_j &= softmax(f_j) \\ c_j &= \textstyle\sum_{i=1}^{T_e} \alpha_{ij} h_i \\ p_{vocab} &= softmax(v_2(v_3[s_j \oplus c_j] + b_2) + b3) \end{aligned}$$

where $v_{1,2,3}$, $b_{1,2,3}$, W_h, and W_s are trainable model parameters and $\oplus$ is the concatenation operator. In a simple sequence-to-sequence model with attention, we use p_{vocab} to calculate the cross-entropy loss. However, since p_{vocab} only captures the distribution of words within the vocabulary, this will generate a lot of OOV words during the decoding step. Therefore, a pointer-generator model mitigates this issue by using a switching mechanism which either chooses a word from vocabulary with a certain probability σ or from the original document using the attention distribution with a probability of (1-σ) as follows:

$$\begin{aligned} \sigma_j &= (W_c c_j + W_s s_j + W_x x_j + b_4) \\ p_j^* &= \sigma_j p_{vocab} + (1 - \sigma_j) \textstyle\sum_{i=1}^{T_e} \alpha_{ij} \end{aligned} \tag{3.2}$$

where W_c, W_x, and b_4 are trainable model parameters and if a word x_j is OOV, then $p_{vocab} = 0$ and the model will rely on the attention values to select the right token. Once the final probability is calculated using Eq. (3.2), the Cross-Entropy (CE) loss is used to calculate the loss as follows:

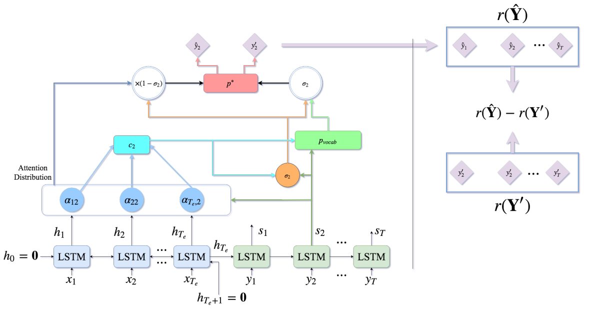


Figure 2: Pointer-generator w. self-critic policy gradient

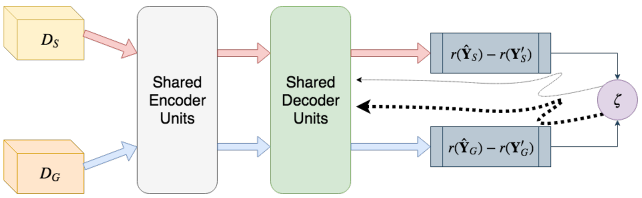
$$\mathcal{L}_{CE} = -\sum_{t=1}^{T} \log p_\theta^*(y_t | e(y_{t-1}), s_t, c_{t-1}, \mathbf{X}) \tag{3.3}$$

where θ shows the training parameters and $e(.)$ returns the word embedding of a specific token. However, as mentioned in Section 2, one of the main problems with cross-entropy loss is the exposure bias [2, 21] which occurs due to the inconsistency between the decoder input during training and test. A model that is trained using only CE loss, does not have the generalization power required for transfer learning, since this model is not trained to generate samples from its own distribution and heavily relies on the ground-truth input. Thus, if the distribution of input data changes (which can likely happen during transfer learning on another dataset), the trained model will have to essentially re-calibrate every transferred layer to achieve a good result on the target dataset. To avoid these problems, we propose a reinforcement learning framework which slowly removes the dependency of model training on the CE loss and increases the reliance of the model on its own output.

3.2.3 Reinforcement Learning Based Objective In RL training, the focus is on minimizing the negative expected reward rather than directly minimizing the CE loss. This allows the framework to not only use the model's output for training itself, but also helps in training the model based on the metric that is used during decoding (such as ROUGE). To achieve this, during RL training, the following objective is minimized:

(3.4)
$$\text{minimize } \mathcal{L}_{RL} = -\mathbb{E}_{y'_1, \cdots, y'_T \sim p_\theta^*(y'_1, \cdots, y'_T)}[r(y'_1, \cdots, y'_T)]$$

[2] For instance, word "is" could have index 1 in the first dataset while it could have index 10 in the second dataset.



D_S
D_G
Shared Encoder Units
Shared Decoder Units
$r(\hat{\mathbf{Y}}_S) - r(\mathbf{Y}'_S)$
$r(\hat{\mathbf{Y}}_G) - r(\mathbf{Y}'_G)$
ζ

Figure 3: The proposed TransferRL framework. The encoder and decoder units are shared between the source (D_S) and target datasets (D_G).

where $y'_1, \cdots, y'_T$ are sample tokens drawn from the output of the policy (p_θ), i.e., $p^*_1, \cdots, p^*_T$. In practice, we usually sample only one sequence of tokens to calculate this expectation. Hence, the derivative of the above loss function is given as follows:

(3.5)
$$\nabla_\theta \mathcal{L}_{RL} = - \mathop{\mathbb{E}}_{y'_1,\cdots,y'_T \sim p^*_\theta} [\nabla_\theta \log p^*_\theta(y'_1, \cdots, y'_T) r(y'_1, \cdots, y'_T)]$$

This minimization can be further improved by adding a baseline reward. In text summarization, the baseline reward could either come from a separate network called critic network [4] or could be the reward from a sequence coming from greedy selection over p^*_t [21]. In this work, we consider the greedy sequence as the baseline. In summary, the objective that we minimize during RL training is as follows:

$$\begin{aligned} \mathcal{L}_{RL} = \; & \textstyle\sum_t - \log p^*_\theta(y_t | y'_{t-1}, s_t, c_{t-1}, \mathbf{X}) \times \\ & \Big(r(\hat{y}_1, \cdots, \hat{y}_T) - r(y'_1, \cdots, y'_T) \Big) \end{aligned} \quad (3.6)$$

where $\hat{y}_t$ represents the greedy selection at time t. This model is also known as *self-critic policy gradient* since the model uses its own greedy output to create the baseline. Moreover, the model uses the sampled sentence as the target for training rather than the ground-truth sentence. Therefore, given this objective, the model focuses on samples that do better than greedy selection during training while penalizing those which do worse than greedy selection.

3.2.4 Transfer Reinforcement Learning Although a model that is trained using Eq. (3.6) does not suffer from exposure bias, it can still perform poorly in a transfer learning setting. This is mostly due to the fact that, the model in this setup, is still being trained using the distribution from the source dataset and once transferred to the target dataset, the model output still tries to generate samples according to the distribution of the source dataset. Therefore, we need a model that not only remembers the distribution of the source dataset but also tries to learn and adapt to the distribution of the target dataset. The overall RL-based framework that is being proposed in this paper is shown in Fig. 3. At each step, we select a random mini-batch from D_S and D_G and feed them to the shared encoder units and decoder starts decoding for each mini-batch. Once the decoding is completed, the model generates a sentence based on the greedy selection and another by sampling from the output distribution. Finally, we calculate the TransferRL loss according to Eq. (3.7) and back-propagate the error according to the trade-off parameter ζ. The thick and thin dashed lines in this plot shows the effect of ζ on the extent to which the model needs to rely on either D_S or D_G for back-propagating the error.

Let us consider sequences drawn from greedy selection and sampling from the source dataset D_S and the target dataset D_G as $\hat{\mathbf{Y}}_S$, $\mathbf{Y}'_S$, $\hat{\mathbf{Y}}_G$, and $\mathbf{Y}'_G$, respectively. We will define the transfer loss function using these variables as follows:

(3.7)
$$\mathcal{L}_{TRL} = - \textstyle\sum_{t=1}^{T} \Bigg(\begin{aligned} & (1-\zeta) \log p^*_\theta(y^S_t | \mathbf{U}_S) \Big(r(\hat{\mathbf{Y}}_S) - r(\mathbf{Y}'_S) \Big) + \\ & \zeta \log p^*_\theta(y^G_t | \mathbf{U}_G) \Big(r(\hat{\mathbf{Y}}_G) - r(\mathbf{Y}'_G) \Big) \end{aligned} \Bigg)$$

where $\mathbf{U}_S = \{e_S(y'_{S,t-1}), s_t, c_{t-1}, \mathbf{X}_S\}$, $\mathbf{U}_G = \{e_S(y'_{G,t-1}), s_t, c_{t-1}, \mathbf{X}_G\}$, and $\zeta \in [0, 1]$ controls the trade-off between self-critic loss of the samples drawn from the source dataset and the target dataset. Therefore, a $\zeta = 0$ means that we train the model only using the samples from the source dataset, while $\zeta = 1$ means that model is trained only using samples from the target dataset. As seen in Eq. (3.7), the decoder state s_t and the context vector c_{t-1} are shared between the source and target samples. Moreover, we use shared embedding trained on the source dataset, $e_S(.)$ for both datasets while the input data given to the encoder, i.e., X_S and X_G, come from source and target datasets.

In practice, RL objective loss only activates after a good pre-trained model is obtained. We follow the same practice and first pre-train the model using the source dataset and then activate the transfer RL loss in Eq. (3.7) by combining this loss to CE loss in Eq. (3.3) using the parameter $\eta \in [0, 1]$ as follows:

$$\mathcal{L}_{Mixed} = (1 - \eta)\mathcal{L}_{CE} + \eta \mathcal{L}_{TRL} \quad (3.8)$$

4 Experimental Results

We performed various sets of experiments to understand the dynamics of transfer learning and to investigate the best practice for obtaining a generalized model for text summarization. In this section, we discuss some of the insights that we gained through our experiments. All evaluations are done using ROUGE 1, 2, and L F-scores on the test data. All our ROUGE scores have a 95% confidence interval of ±0.25 as reported by the official

Table 2: Basic statistics for the datasets used in our experiments.

	Newsroom	**CNN/DM**	**DUC'03**	**DUC'04**
# Train	994,001	287,226	0	0
# Eval	108,312	13,368	0	0
# Test	108,655	11,490	624	500
Avg. # summary sentences	1.42	3.78	4	4
Avg. # words in summary	21	14.6	11.03	11.43

ROUGE script[3]. Similar to the multi-task learning frameworks such as DecaNLP [16], we use a metric for comparing the result of transfer learning on various datasets by taking the average score of each metric over these datasets. In addition, we also introduce a weighted average score which takes into account the size of each dataset as the weight for averaging the values[4].

4.1 Datasets We use four widely used datasets in text summarization for our experiments. The first two datasets are Newsroom [9] and CNN/Daily Mail [10] which are used for training our models, while the DUC 2003 and DUC 2004 datasets are only used to test the generalization capability of each model. Table 2 shows some of the basic statistics of these datasets. In all these datasets, a news article is accompanied by 1 to 4 human-written summaries and, therefore, will cover a wide range of challenges for transfer learning. For instance, a model that is trained on Newsroom dataset will most likely generate only one long summary sentence, while for the CNN/DM dataset, the model is required to generate up to four smaller summary sentences. For all experiments, we either use Newsroom as D_S and CNN/DM as D_G or vice-versa.

4.2 Training Setup For each experiment, we run our model for 15 epochs during pre-training and 10 epochs during transfer process and an extra 2 epochs for the coverage mechanism. We use a batch size of 48 during training, the encoder reads the first 400 words, and decoder generates a summary with 100 words. Both encoder and decoder units have a hidden size of 256 while the embedding dimension is set to 128 and we learn the word embedding during training. For all models, we used the top 50K words in each dataset as the vocabulary and during test we use beam search of size 4. We use AdaGrad to optimize all models with an initial learning rate of $\gamma_0 = 0.15$ during pre-training and $\gamma_0 = 0.001$ during RL and coverage and linearly decrease this learning rate based on the epoch numbers as $\gamma_t = \gamma_0/epoch$. Moreover, ζ is set to zero at the start of RL training and is increased linearly so that it gets to 1 by the end of training. During RL training, we use scheduled sampling with sampling probability equal to the ζ value. We use the RLSeq2Seq [11] framework to build our model and the code is publicly available[5].

4.3 Effect of Dataset Size We will now discuss some of the insights we gained throughout our study starting with understanding the effect of the data size for pre-training. According to our experiments (as shown in Table 4), on average, a model that is trained using the Newsroom dataset as the source dataset D_S has much better performance than models that use CNN/DM as the D_S in almost all configurations[6]. This is not a surprising result since deep neural networks are data hungry models and typically work the best when provided with a large number of samples. The first experiment in Table 3 and Table 4 uses only Newsroom dataset for training the model and not surprisingly it performs good on this dataset, however as discussed earlier, its performance on other datasets is poor.

4.4 Common Vocabulary As mentioned in Section 3, one way to avoid the excessive OOV words in transfer learning between two datasets is to use a common vocabulary between D_S and D_G and train a model using this common vocabulary set. Although a model trained using this set of vocabulary could perform well on these two datasets, it still suffers from poor generalization to other unseen datasets. To demonstrate this, we combine all articles in CNN/DM and Newsroom training datasets to create a single unified dataset (C+N in Table 3 and Table 4) and train a model using CE loss in Eq. (3.3) and the common set of vocabulary. The result of this experiment is shown as experiment 2 in Table 3 and Table 4. As shown here, by comparing these results to experiment 1, we see that combining these two datasets will decrease the performance on Newsroom, DUC'03, and DUC'04 test datasets but improves the performance for CNN/DM test data. Moreover, by comparing the generalization ability of this method on DUC'03 and DUC'04 datasets, we see that it performs up to 2% worse than the proposed method. This is also shown by comparing the average scores and weighted average scores of our proposed model with this model. On average, our method improves up to 4% compared to this method according to the R_1 weighted average score.

[3] `https://pypi.org/project/pyrouge/`

[4] Note that, due to page limitation, some of the results are presented in the Arxiv version of the paper `https://arxiv.org/abs/1810.06667`.

[5] Source code: `www.github.com/yaserkl/TransferRL`

[6] Due to space constraint, we only report the results from this setup and refer the readers to the Arxiv version of the paper

Table 3: Results on Newsroom, CNN/DM, DUC'03, and DUC'04 test data. D_S shows the dataset that is used during pre-training and D_G is our target dataset. **N** stands for Newsroom and **C** stands for CNN/DM dataset. The method column shows whether we use CE loss, transferring layers (TL), or TransferRL (TRL) loss during training. We use coverage mechanism for all experiments. The result from the proposed method is shown with $\star$.

#	D_S	D_G	**Method**	**Newsroom**			**CNN/DM**			**DUC'03**			**DUC'04**		
				R_1	R_2	R_L	R_1	R_2	R_L	R_1	R_2	R_L	R_1	R_2	R_L
1	**N**		CE Loss	36.16	24.33	32.87	33.58	12.76	29.72	28.03	9.15	24.75	29.85	10.3	26.7
2	**C+N**		CE Loss	30.26	17.68	27.03	**38.23**	**16.31**	**34.66**	26.71	7.81	24.13	27.96	8.25	25.25
3	**N**	**C**	TL	35.37	23.45	32.07	34.51	13.49	30.61	28.19	9.34	24.96	29.83	9.98	26.66
4	**N**	**C**	TRL*	**36.5**	**24.77**	**33.25**	35.24	13.56	31.33	**28.46**	**9.65**	**25.45**	**30.45**	**10.63**	**27.42**

Table 4: Normalized and weighted normalized ROUGE F-Scores for Table 3.

#	D_S	D_G	**Method**	**Avg. Score**			**Weighted Avg. Score**		
				R_1	R_2	R_L	R_1	R_2	R_L
1	**N**		CE Loss	31.91	14.14	28.51	35.58	21.73	32.16
2	**C+N**		CE Loss	30.79	12.51	27.77	32.04	17.36	28.74
3	**N**	**C**	TL	31.98	14.07	28.58	35.17	21.21	31.74
4	**N**	**C**	TRL*	**32.66**	**14.65**	**29.36**	**36.21**	**22.25**	**32.81**

4.5 Transferring Layers In this experiment, we discuss the effect of transferring different layers of a pre-trained model for transfer learning. In the pointer generator model described in Section 3, the embedding matrix, the encoder and decoder model parameters are among layers that we can use for transfer learning. For this experiment, we pre-train our model using D_S and during transfer learning, we replace D_S with D_G and continue training of the model with CE loss. As shown in Tables 3 and 4 this way of transferring network layers provides a strong baseline for comparing the performance of our proposed method. This shows that even a simple transferring of layers could provide enough signals for the model to adapt itself to the new data distribution. However, as discussed earlier in Section 3, this way of transfer learning tends to completely forget the pre-trained model distribution and entirely changes the final model distribution according to the dataset that is used for fine-tuning. This effect can be observed in Table 3 by comparing the result of experiments 1 and 3. As shown in this table, after transfer learning the performance drops on Newsroom test dataset (from 36.16 to 35.37 based on R_1) while it increases on CNN/DM dataset (from 33.58 to 34.51 based on R_1). However, since our proposed method tries to remember the distribution of the pre-trained model (through ζ parameter) and slowly changes the distribution of the model according to the distribution coming from the target dataset, it performs better than simple transfer learning on these two datasets. This is shown by comparing the result in experiments 3 and 4 in Table 3, which shows that our proposed model performs better than naïve transfer learning in all test datasets.

4.6 Effect of Zeta As mentioned in Section 3, the trade-off between emphasizing the training to samples drawn from D_S or D_G is controlled by the hyper-parameter ζ. To see the effect of ζ on transfer learning, we clip the ζ value at 0.5 and train a separate model using this objective. Basically, a $\zeta = 0.5$ means that we treat the samples coming from source and target datasets equally during training. Therefore, for these experiments, we start the ζ from zero and increase it linearly till the end of training but clip the ζ value at 0.5. Table 5 shows the result of this experiment. For simplicity sake, we provide the result of our proposed model achieved from not clipping ζ along with these results. By comparing the results from these two setups, we can see that, on average, increasing the value of ζ to 1.0 will yield better results than clipping this value at 0.5. For instance, according to the average and weighted average score there is an increase of 0.7% in ROUGE-1 and ROUGE-L scores when we do not clip the ζ at 0.5. By comparing the CNN/DM R_1 score in this table, we see that clipping the ζ value will definitely hurt the performance on D_G since the model shows equal attention to the distribution coming from both datasets. On the other hand, the surprising component here, is that, by avoiding ζ clipping, the performance on the source dataset also increases [7].

4.7 Transfer Learning on Small Datasets As discussed in Section 1, transfer learning is good for situations where the goal is to do summarization on a dataset with little or no ground-truth summaries. For

[7] For $\zeta \in (0.5, 1)$, we have only seen small improvement in the results and hence we omitted them from reporting in this section.

Table 5: Result of TransferRL after clipping ζ at 0.5 and $\zeta = 1.0$ on Newsroom, CNN/DM, DUC'03, and DUC'04 test data along with the average and weighted average scores.

ζ	**0.5**			**1.0**		
	R_1	R_2	R_L	R_1	R_2	R_L
Newsroom	36.06	24.23	32.78	**36.5**	**24.77**	**33.25**
CNN/DM	33.7	12.83	29.81	**35.24**	**13.56**	**31.33**
DUC'03	28.3	9.54	25.04	**28.46**	**9.65**	**25.45**
DUC'04	29.88	10.23	26.8	**30.45**	**10.63**	**27.42**
Avg. Score	31.98	14.2	28.6	**32.66**	**14.65**	**29.36**
Weighted Avg. Score	35.52	21.66	32.11	**36.21**	**22.25**	**32.81**

this purpose, we conducted another set of experiments to test our proposed model on transfer learning using DUC'03 and DUC'04 datasets as our target dataset, i.e., D_G. For these experiments, we randomly pick 20% of each dataset as our train set and 10% as validation dataset and the rest of the dataset as our test data. This will generate 124 and 100 articles as our train dataset for DUC'03 and DUC'04, respectively. Similar to other experiments in this paper, we use CNN/DM and Newsroom as D_S[8] while using DUC'03 and DUC'04 as D_G during transfer learning. Due to the size of these datasets, the models are trained only for 3000 iterations during fine-tuning and best model is selected according to the validation set. Tables 6 and 7 show the results of this experiment. As shown in these tables, for DUC'03, when we simply transfer network layers, it performs slightly better (not statistically higher according to the 95% confidence interval) than our proposed model, however, our proposed model will achieve a far better result on DUC'04. As shown in these tables, the results achieved from fine-tuning a pre-trained model using these datasets is very close to the ones achieved in Table 3 and in the case of DUC'04 dataset, our proposed method in Table 3 achieves even better results than the ones shown in Table 7. This shows the ability of our proposed framework in generalizing to unseen datasets. Note that, unlike these experiments, the proposed model in Table 3 has no information about the data distribution of DUC'03 and DUC'04 and still performs better on these datasets.

4.8 Other Generalized Models We also compare the performance of our proposed model with some of the recent works in multi-task learning. In text summarization, the DecaNLP [16] and Tensor2Tensor [33] are two of the most recent frameworks that use multi-task learning. Following the setup in these works, we focus on the models that are trained using CNN/DM dataset and report the average ROUGE 1, 2, and L F-scores for our best performing model. Table 8 compares the result of our proposed method with these methods.

Table 6: Result of transfer learning methods using Newsroom for pre-training and DUC'03 for fine-tuning. The underlined result shows that the improvement from TL is not statistically significant compared to the proposed model.

D_S	D_G	**Method**	R_1	R_2	R_L
N	**DUC'03**	TL	28.9	9.73	25.54
N	**DUC'03**	TRL*	28.76	9.5	25.39

Table 7: Result of transfer learning using Newsroom for pre-training and DUC'04 for fine-tuning.

D_S	D_G	**Method**	R_1	R_2	R_L
N	**DUC'04**	TL	27.68	9.09	25.29
N	**DUC'04**	TRL*	**29.54**	**9.99**	**26.56**

Table 8: Comparing our best performing model with the state-of-the-art multi-task learning frameworks on CNN/DM dataset and according to the average of ROUGE 1, 2, and L F-scores. The result with $*$ is the same as reported in original paper.

	DecaNLP [16]	**Tensor2Tensor [33]**	**Proposed Model**
Average ROUGE	25.7*	27.4	**31.12**

5 Conclusion

In this paper, we tackled the problem of transfer learning in text summarization. We studied this problem from different perspectives through transfer of network layers from a pre-trained model to proposing a reinforcement learning framework which borrows insights from self-critic policy gradient and offers a systematic mechanism that creates a trade-off between the amount of reliance on the source or target dataset during training. Through an extensive set of experiments, we showed the generalization power of the proposed model on unseen test datasets and reaching state-of-the-art results on such datasets. To the best of our knowledge, this is the first work that studies transfer learning in text summarization and offers a solution that beats state-of-the-art models and generalizes well to unseen datasets.

Acknowledgments

This work was supported in part by the US National Science Foundation grants IIS-1619028, IIS-1707498 and IIS-1838730.

References

[1] J. Ba and R. Caruana. Do deep nets really need to be deep? In *NIPS*, pages 2654–2662, 2014.

[8] The CNN/DM results are excluded due to space constraints, the reader can refer to Arxiv version of the paper.

[2] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, pages 1171–1179, 2015.

[3] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *NIPS*, pages 523–531, 2016.

[4] Y.-C. Chen and M. Bansal. Fast abstractive summarization with reinforce-selected sentence rewriting. In *ACL*, volume 1, pages 675–686, 2018.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655, 2014.

[6] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. In *NIPS*, pages 1087–1098, 2017.

[7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[8] S. Gehrmann, Y. Deng, and A. M. Rush. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*, 2018.

[9] M. Grusky, M. Naaman, and Y. Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *NAACL-HLT*, 2018.

[10] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701, 2015.

[11] Y. Keneshloo, T. Shi, N. Ramakrishnan, and C. K. Reddy. Deep reinforcement learning for sequence to sequence models. *arXiv:1805.09461*, 2018.

[12] W. Kryściński, R. Paulus, C. Xiong, and R. Socher. Improving abstraction in text summarization. *arXiv preprint arXiv:1808.07913*, 2018.

[13] Z. Li, X. Jiang, L. Shang, and H. Li. Paraphrase generation with deep reinforcement learning. *arXiv preprint arXiv:1711.00279*, 2017.

[14] B. Y. Lin and W. Lu. Neural adaptation layers for cross-domain named entity recognition. In *EMNLP*, pages 2012–2022, 2018.

[15] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Proc. of Workshop on Text Summarization Branches Out*, 2004.

[16] B. McCann, N. S. Keskar, C. Xiong, and R. Socher. The natural language decathlon: Multitask learning as question answering. *arXiv:1806.08730*, 2018.

[17] R. Nallapati, F. Zhai, and B. Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081, 2017.

[18] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *SIGNLL*, pages 280–290, 2016.

[19] S. Narayan, S. B. Cohen, and M. Lapata. Ranking sentences for extractive summarization with reinforcement learning. In *NAACL-HLT*, 2018.

[20] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318, 2002.

[21] R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[22] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[23] E. M. Ponti, I. Vulić, G. Glavaš, N. Mrkšić, and A. Korhonen. Adversarial propagation and zero-shot cross-lingual transfer of word vector specialization. In *EMNLP*, pages 282–293. ACL, 2018.

[24] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2016.

[25] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, pages 379–389, 2015.

[26] D. S. Sachan, P. Xie, and E. P. Xing. Effective use of bidirectional language modeling for medical named entity recognition. *arXiv, arXiv:1711.07908*, 2017.

[27] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, pages 1842–1850, 2016.

[28] A. See, P. J. Liu, and C. D. Manning. Get to the point: Summarization with pointer-generator networks. In *ACL*, volume 1, pages 1073–1083, 2017.

[29] T. Semwal, P. Yenigalla, G. Mathur, and S. B. Nair. A practitioners' guide to transfer learning for text classification using convolutional neural networks. In *SDM*, pages 513–521. SIAM, 2018.

[30] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy. Neural abstractive text summarization with sequence-to-sequence models. *arXiv preprint arXiv:1812.02303*, 2018.

[31] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087, 2017.

[32] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *CVPR*, volume 1, page 4, 2017.

[33] A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, Ł. Kaiser, N. Kalchbrenner, N. Parmar, et al. Tensor2tensor for neural machine translation. *arXiv:1803.07416*, 2018.

[34] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328, 2014.

[35] Q. Zhou, N. Yang, F. Wei, S. Huang, M. Zhou, and T. Zhao. Neural document summarization by jointly learning to score and select sentences. In *ACL*, pages 654–663. ACL, 2018.