Deep Headline Generation for Clickbait Detection

Kai Shu*, Suhang Wang†, Thai Le†, Dongwon Lee†, and Huan Liu*
*Arizona State University, {kai.shu, huan.liu}@asu.edu
†Penn State University, {thai.le, dlee}@ist.psu.edu, szw494@psu.edu

***Abstract*—Clickbaits are catchy social posts or sensational headlines that attempt to lure readers to click. Clickbaits are pervasive on social media and can have significant negative impacts on both users and media ecosystem. For example, users may be misled to receive inaccurate information, or fall into click-jacking attacks. Similarly, media platforms could lose readers' trust and revenues due to the prevalence of clickbaits. To computationally detect such clickbaits on social media using supervised learning framework, one of the major obstacles is the lack of large-scale labeled training data, due to laborious and costly labeling. With the recent advancements in deep generative models, to address this challenge, we propose to generate *synthetic* headlines with specific styles and explore their utilities to help improve clickbait detection. In particular, we propose to generate stylized headlines from original documents with *style transfer*. Furthermore, as it is non-trivial to generate stylized headlines due to several challenges such as the discrete nature of texts and the requirements of preserving semantic meaning of the document while achieving style transfer, we propose a novel solution, named as *Stylized Headline Generation (SHG)*, that can not only generate readable and realistic headlines to enlarge original training data, but also helps improve the classification capacity of supervised learning. The experimental results on real-world datasets demonstrate the effectiveness of SHG on generating high-quality and high-utility headlines for clickbait detection.**

***Index Terms*—Data augmentation, deep generative model, clickbait detection**

I. Introduction

People nowadays often seek out and consume information from social media due to its accessibility, low cost and fast dissemination. However, the prosperity of online journalism also produces large amounts of *clickbaits*, which refer to catchy social posts or sensational headlines that attempt to lure readers to click by creating an information gap in their minds and arousing their curiosities. A recent study reported that clickbaits are prevalent in both mainstream and alternative media venues, with clickbaits are seen as much as 33.54% on mainstream media and 39.26% on alternative media sites[1]. Clickbait is growing fast in volume and it can have many negative societal impacts. Firstly, clickbaits may contain sensational and inaccurate information to mislead readers and spread fake news [1]. Secondly, news media may confront with losing readers' trust and depleting the brand value [2]; and thirdly, clickbaits may be used to perform click-jacking attacks by redirecting users to phishing websites to steal their personal information [3]. Thus, it is critical to detect clickbaits on social media.

[1]http://newslab.org/clickbait-headlines-grow-mainstream-media/

Existing approaches to detect clickbaits mainly focus on extracting discriminative features such as hand-crafted linguistic features [4], or constructing effective models such as deep neural networks [5], [2]. However, these methods may face following limitations: (i) the datasets are often limited in their scale and (ii) the imbalanced distribution of labeled instances, due to the expensive cost to obtain adequate training examples with manual labels. One way to build an effective clickbait detection system is to generate synthetic headlines with specific style labels and exploit its utility to improve clickbait detection.

Recently, deep generative models such as Generative Adversarial Nets (GANs) and Variational Auto-Encoder (VAE) with Recurrent Neural Networks (RNNs) as the generator have shown promising results in labeled text generation [6], [7]. For example, Shen *et al.* propose a cross-aligned auto-encoder which can transfer the style of sentences and preserve the content of original sentences. In [6], a VAE is utilized to encode sentences into disentangled content representation, which is further combined with a target style to generate headlines. Despite the success of existing deep learning based labeled text generation methods, the majority of existing methods mainly takes a label and/or a headline as input to generate labeled or stylized headline, which may not be applicable to headline generation because such process is nontrivial. In addition to generating labeled headlines so as to augment the dataset for training better classifiers, the generated headlines should also preserve certain information of the documents. Such preservation is important because: i) it provides necessary information to guide the generation of headlines; and ii) generating content preserved headlines makes it possible to suggest a non-clickbait headline to readers after we detect a clickbait. Thus, it is important to propose a generative model that can generate labeled headlines that also preserve document contents, which has the potential to augment data for clickbait detection and also suggest non-clickbait headlines if clickbaits are detected.

Therefore, in this paper, we study a novel problem of stylized headline generation from original documents for clickbait detection. To this end, we build our headline generation framework through a coherent process which consists of a generator learning component and a discriminator learning component. The goal of the generator learning is twofold: i) a *document auto-encoder* for extracting latent representations of the original document's content; and ii) a *headline generator* for generating headlines with specific styles guided by the latent content representations and style vectors indicating the style labels. The discriminators are utilized to regularize the generating process by incorporating necessary constraints. First, we incorporate a *pair discriminator* to ensure the relationships of documents and their correspondent headlines are preserved. Second, we introduce a *style discriminator* to maximize the differentiability of styles for original headlines and generated headlines. Third, we use a *transfer discriminator* for adversarial training to ensure that the generated headlines and transferred headlines that have the same styles are similar in a distribution, where the headline generator is trained to fool the transfer discriminator, so that it cannot easily differentiate original headlines and generated headlines in a distribution.

In essence, we investigate the following challenges: 1) How to generate realistic and readable headlines from original document to improve clickbait detection; 2) How to generate headlines that can preserve the content of documents and transfer the style of headlines? Our solutions to these challenges results in a novel framework called SHG for generating synthetic stylized headlines to improve clickbait detection. Our main contributions are summarized as follows:

- We study a novel problem of generating stylized headlines for clickbait detection problem;
- We provide a principled way to generate headlines from original articles that can transfer headline styles accurately while preserving article content; and
- We conduct extensive experiments on real-world datasets to demonstrate the effectiveness in generating readable, discriminative, consistent and useful headlines using the proposed framework.

The rest of the paper is organized as follows. In Section II, we formally define the problem of headline generation with style transfer for clickbait detection. In Section III, we introduce the proposed SHG framework and as well as its training strategy in details. We present empirical evaluations to justify the effectiveness of SHG with the discussion in Section IV. In Section V, we briefly review the related work. Finally, we conclude with future work in Section VI.

II. Problem Formulation

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ denote the set of documents, $\mathcal{H} = \{h_1, h_2, \ldots, h_m\}$ denote the set of corresponding headlines, and $\mathcal{Y}^{(L)} = \{y_1^L, y_2^L, \ldots, y_m^L\}$ are the given style labels, indicating whether the headline h_i is clickbait ($y_i^L = 1$) or non-clickbait ($y_i^L = 0$). Each document x_i and its headline h_i are composed by a sequence of words from a fixed vocabulary $\mathcal{V}$ with size K, i.e., $x_i = (x_i^1, x_i^2, \ldots, x_i^M)$ and $h_i = (h_i^1, h_i^2, \ldots, h_i^N)$. We use the set of tuples $\mathcal{S} = \{(x_i, h_i) | i = 1, 2, \ldots, m\}$ to denote the correspondent (document, headline) pairs. We denote $\mathbf{x}_i \in \mathbb{R}^{l\times 1}$ as the original document feature vector, $\mathbf{z} \in \mathbb{R}^{d\times 1}$ as the latent content representation of the document. $\mathbf{y}_i^L \in \mathbb{R}^{k\times 1}$ is the label representation of the original headline h_i, and $\mathbf{y}_i^U \in \mathbb{R}^{k\times 1}$ is the label representation for the generated headline o_i. Then, in this paper, we study the following problem:

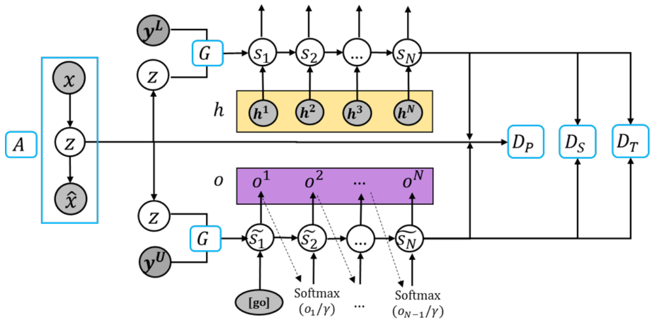


Fig. 1: The proposed framework of generating headlines from documents with style transfer. The framework consists of five components: a document auto-encoder A, a headline generator G, a transfer discriminator D_T, a style discriminator D_S, and a pair discriminator D_P.

Problem 1. *Given a set of tuples $\mathcal{S}$, learn a generator f that can generate stylized headlines given a document and a style label, i.e., $o_i = f(x_i, y_i)$.*

III. Stylized Headline Generation Framework

In this section, we detail the framework of stylized headline generation. *Note that the training data is inherently non-parallel, i.e., for each document, we only have one headline that is either clickbait or non-clickbait.* In other words, for a given pair of document and headline (x_i, h_i), we do not have a parallel headline that is also associated with x_i but have opposite style label with h_i. Such non-parallel data makes it more difficult to train a stylized headline generator because we lack the parallel headlines to guide the learning process. To tackle such a challenge, we propose a framework to generate both click-bait and non-clickbait with style transfer. We illustrate the entire model as in Figure 1. It consists of five major components: a document auto-encoder, a headline generator, a transfer discriminator, a style discriminator and a pair discriminator. In general, the document encoder A aims to learn the content representation of a document by minimizing the reconstruction errors. The headline generator G can generate headlines from content and style representations with style transfer from the original style to the target style. The style discriminator D_S learns a style classifier to guide the style learning process. The transfer discriminator D_T aims to ensure the original headlines and transferred headlines with the same styles are similar as a distribution. The pair discriminator D_P ensures that the learned representations can maintain the consistency of the pair relationship of documents and headlines.

A. Generator Learning

1) Document Auto-encoder: To generate a headline with a specific style for the original document, we need to ensure that the content in the document is preserved in the generated headline. Thus, we first demonstrate how to extract the content representation for the document. Auto-encoder has been widely utilized for text generation and has shown to be effective recently [8]. So we also use an auto-encoder A to extract content representation $\mathbf{z}$ from document $x = \{x^1, x^2, ..., x^M\}$, where x^i is the i_{th} word in the document. Let $E_A : \mathcal{X} \rightarrow \mathcal{Z}$ be an encoder that can infer the content representation $\mathbf{z}$ for a given document x, and $D_A : \mathcal{Z} \rightarrow \mathcal{X}$ be a decoder that reconstruct the document from the learned latent representation of documents.

We apply recurrent neural networks (RNN) as the encoder E_A. An RNN can learn a probability distribution over a sequence by being trained to predict the next symbol in a sequence. A RNN consists of a hidden state S and an optional output which operates on a word sequence $x = \{x^1, x^2, ..., x^M\}$. At each time step t, the hidden state $\mathbf{s}_t$ of RNN is updated by,

$$\mathbf{s}_t = f_{enc}(\mathbf{s}_{t-1}, x^t) \tag{1}$$

After reading the end of the document, the last hidden state of the RNN is used as the representation vector $\mathbf{z}$ of the whole document. We employ the gated recurrent unit (GRU) as the cell type to build the RNN, which is designed in a manner to have a more persisted memory [9]. Let θ_e denote the parameters for encoder, then we have,

$$\mathbf{z} = E_A(x, \theta_E) \tag{2}$$

The decoder takes $\mathbf{z}$ as the input to start the generation process. We use another RNN to build the decoder D_A to generate the output word sequence $\hat{x} = \{\hat{x}^1, \ldots, \hat{x}^M\}$. At each time step t, the hidden state of the decoder is computed by,

$$\mathbf{s}_t = f_{dec}(\mathbf{s}_{t-1}, \hat{x}^t) \tag{3}$$

where $\mathbf{s}_0 = \mathbf{z}$. The word at time step t is predicted by a softmax classifier as follows,

$$\hat{\mathbf{y}}_t = \text{softmax}(\mathbf{W}^{(S)}\mathbf{s}_t) \tag{4}$$

where softmax$(\cdot)$ is a softmax activation function. $\hat{\mathbf{y}}_t \in \mathbb{R}^{|\mathcal{V}|}$ is a probability distribution over the vocabulary, and $\mathbf{W}^{(S)} \in \mathbb{R}^{|\mathcal{V}| \times (d+k)}$ with $d+k$ as the dimension of the hidden state in each layer. The probability of choosing word v_j ($v_j \in \mathcal{V}$) as the output is,

$$p(\hat{x}_t = v_j | \hat{x}_{t-1}, \hat{x}_{t-2}, \ldots, \hat{x}_1) = \hat{\mathbf{y}}_{t,j} \tag{5}$$

Thus, the overall probability of generating an output sequence $\hat{x}$ giving the input document x is defined as follows,

$$p(\hat{x}|x; \theta_d) = \prod_{t=1}^{M} p(\hat{x}_t | \hat{x}_{t-1}, \hat{x}_{t-2}, \ldots, \hat{x}_1, \mathbf{z}, \theta_d) \tag{6}$$

The two components of the proposed auto-encoder are jointly trained to minimize the negative conditional log-likelihood for all documents,

$$\mathcal{L}_{rec}(\theta_e, \theta_d) = -\sum_{i=1}^{m} \log p(\hat{x}_i | x_i; \theta_d, \theta_e) \tag{7}$$

where θ_e and θ_d are the set of model parameters of the encoder and decoder, respectively. Following traditional setting, we set the output document $\hat{x}$ as the reverse sequence of original document x.

2) Headline Generator: The headline generator G takes the document content representation $\mathbf{z}$ and the style label representation as input and generate headline with the corresponding style. For example, if we have an original headline h with style $y^L = 1$ (clickbait), then we want to generate a new headline o with style $y^U = 0$ (non-clickbait). In this case, we can generate a non-clickbait headline giving the document with the original headline as a clickbait.

Given the document content representations $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$, a good headline generator G should be able to: i) reconstruct all the original headlines $\mathcal{H}$; and ii) generate a set of new headlines $\mathcal{O}$ with the opposite styles with original headlines; and iii) ensure that the distributions of the generated headlines and original headlines which have same style should be indistinguishable. To this end, we employ an adversarial training scheme to utilize the generator G and a discriminator D_T (more details in Section III-B2) simultaneously. Besides the reconstruction loss which drives the generator to produce realistic headlines, the discriminator D_T provides extra learning signals which enforce the generator G to produce coherent representations that match the style labels.

However, adversarial training over the discrete samples (words) hinders gradients propagation. To solve this challenge, we propose to apply a deterministic *continuous approximation* [6]. The continuous approximation replaces the sampled token o^t (represented as a one-hot vector) at each step with the probability vector defined as follows:

$$o^t \sim \text{softmax}(o^t/\gamma) \tag{8}$$

which is differentiable w.r.t the parameters of generator G. The probability vector is used as the output at the current step and the input to the next step along the sequence of decision making.

Thus, for each tuple $(x_i, h_i) \in \mathcal{S}$, we have one sequence generated by $G(\mathbf{z}, \mathbf{y}^L)$ teach-forced by the ground-truth sample h, and the other free-running one generated by $G(\mathbf{z}, \mathbf{y}^U)$ using the self-generated by previous continuous approximation as input at each step, resulting a generated headline o. The objective function of the generator G is to optimize the reconstruction error of observed documents and generated headlines, by minimizing the following negative log likelihood:

$$\mathcal{L}_G(\theta_G) = \mathbb{E}_{(x,h)\in\mathcal{S}}[-\log p_G(h|\mathbf{y}^L, \mathbf{z}))] \tag{9}$$

B. Discriminator Learning

We have the following three discriminators aiming to constrain the generated headlines from different perspectives: a *style discriminator*, a *transfer discriminator*, and a *pair discriminator*. The style discriminator can classify the styles of both generated and original headlines to help the generation process. The transfer discriminator can differentiate the original and generated headlines to ensure they have aligned distributions. The pair discriminator describes the process of forcing the label of pairs of documents and headlines are correctly classified.

1) Style Discriminator: The generated headline o and the original headline h should be different in terms of their styles. Thus, we define a style discriminator D_S that aims to assign a correct label of styles for both original headlines and generated headlines, which can further guide the headline generator G through parameter back-propagation.

We aim to learn good representations of headlines with good discriminative capacities to differentiate style labels. We have $y^L \in \{0,1\}$, where $y^L = 0$ indicates the headline is a non-clickbait, and $y^L = 1$ indicates the headline is a clickbait. We use a linear classifier, i.e., softmax classifier, as the style discriminator. Thus, we can predict the style label as follows,

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{s}_N \mathbf{W} + \mathbf{b}) \tag{10}$$

where $\hat{\mathbf{y}} = (\hat{\mathbf{y}}_0, \hat{\mathbf{y}}_1)$ is the predicted probability vector with $\hat{\mathbf{y}}_0$ and $\hat{\mathbf{y}}_1$ indicate the predicted probability of label being 0 and 1, respectively. $\mathbf{s}_N \in \mathbb{R}^{1\times(d+k)}$ is the last hidden state of the generator G for reconstructing the original headline, and $\mathbf{W} \in \mathbb{R}^{(k+d)\times 2}$ is the weight of the softmax function, $\mathbf{b} \in \mathbb{R}^{1\times 2}$ is a bias term, and k is the dimension of hidden state of RNN. Thus, for each original headline, the goal is to minimize the cross-entropy loss function as follows,

$$\mathcal{L}_{D_S}^{(1)}(\mathbf{W}, \mathbf{b}) = -y^L \log(\hat{\mathbf{y}}_1) - (1 - y^L)\log(1 - \hat{\mathbf{y}}_0) \tag{11}$$

Similarly, for each generated headline, we define the following softmax classifier to predict the style label,

$$\hat{\tilde{\mathbf{y}}} = \text{softmax}(\tilde{\mathbf{s}_N} \mathbf{W} + \mathbf{b}) \tag{12}$$

where $\hat{\tilde{\mathbf{y}}} = (\hat{\tilde{\mathbf{y}}}_0, \hat{\tilde{\mathbf{y}}}_1)$ is the predicted probability vector with $\hat{\tilde{\mathbf{y}}}_0$ and $\hat{\tilde{\mathbf{y}}}_1$ indicate the predicted probability of label being 0 and 1 for the generated headline, respectively. For each generated headline, the goal is to minimize the cross-entropy loss function as follows,

$$\mathcal{L}_{D_S}^{(2)}(\mathbf{W}, \mathbf{b}) = -y^U \log(\hat{\tilde{\mathbf{y}}}_1) - (1 - y^U)\log(1 - \hat{\tilde{\mathbf{y}}}_0) \tag{13}$$

Thus, we combine Eqn: 11 and Eqn: 13 and obtain the unified form of cross-entropy loss for both original headlines and generated headlines,

$$\mathcal{L}_{D_S}(\mathbf{W}, \mathbf{b}) = \mathcal{L}_{D_S}^{(1)} + \mathcal{L}_{D_S}^{(2)} \tag{14}$$

2) Transfer Discriminator: In this section, we introduce how to discriminate original data samples with generated data samples using adversarial learning. Following the setting of GANs [10], we add a transfer discriminator D_T, and the generator parameters θ_G are trained to fool the discriminator D_T, so that D_T can not easily differentiate original headlines and generated headlines in a distribution. Thus, the objective of the discriminator D_T is to ensure the generated headlines are as realistic and close as possible to original headlines with the same styles.

Since the original headlines have different styles, their data distributions should also be different. Thus, we propose to use two transfer discriminators $D_T^{(1)}$ and $D_T^{(2)}$ to differentiate whether headlines are original or generated. Specifically, if $D_T^{(1)}$ is to discriminate original headlines with $y^L = 0$ and generated headlines with $y^U = 1$; and $D_T^{(2)}$ is to discriminate original headlines with $y^L = 1$ and generated headlines with $y^U = 0$. We aim to minimize the following negative log likelihood,

$$\begin{aligned}\mathcal{L}_{D_T^{(1)}}(\theta_{D_T^{(1)}}) =& \mathbb{E}_{y^U=1}[-\log(1 - D_T^{(1)}(\tilde{\mathbf{s}}_N))] \\ &+ \mathbb{E}_{y^L=0}[-\log D_T^{(1)}(\mathbf{s}_N)]\end{aligned} \tag{15}$$

$$\begin{aligned}\mathcal{L}_{D_T^{(2)}}(\theta_{D_T^{(2)}}) =& \mathbb{E}_{y^U=0}[-\log(1 - D_T^{(2)}(\tilde{\mathbf{s}}_N))] \\ &+ \mathbb{E}_{y^L=1}[-\log D_T^{(2)}(\mathbf{s}_N)]\end{aligned} \tag{16}$$

where $\theta_{D_T^{(1)}}$ and $\theta_{D_T^{(2)}}$ are the parameters to be inferred for discriminator D_T^1, and discriminator D_T^2. $\mathbf{s}_N$ and $\tilde{\mathbf{s}}_N$ are the learned representations of a real headline and a generated headline. The overall loss is as follows,

$$\mathcal{L}_{D_T} = \mathcal{L}_{D_T^{(1)}}(\theta_{D_T^{(1)}}) + \mathcal{L}_{D_T^{(2)}}(\theta_{D_T^{(2)}}) \tag{17}$$

3) Pair Discriminator: During the headline generation process, one goal is to ensure that the correspondences of documents and headlines are maintained. Thus, we need to ensure that the representations of both original headlines and generated headlines are correctly aligned to the original documents. For simplicity of explanation, we only consider the case of original headline and the document. We first define a proximity function which can capture the similarity between the representations of a headline h_i and a document x_j as follows,

$$p(h_i, x_j) = \frac{1}{1 + \exp(-\mathbf{s}^{(i)}\mathbf{Q}\mathbf{z}^{(j)})} \tag{18}$$

where $\mathbf{s}^{(i)} \in \mathbb{R}^{1\times k}$ and $\mathbf{z}^{(j)} \in \mathbb{R}^{t\times 1}$ are the latent representations of headline h_i and x_j. $\mathbf{Q} \in \mathbb{R}^{k\times t}$ is a linear mapping matrix that connects $\mathbf{s}^{(i)}$ and $\mathbf{z}^{(j)}$. Thus, we introduce a discriminator D_P, which can handle following situations: i) for $(h_i, x_i) \in \mathcal{S}$, we should maximize the proximity value; ii) for $(h_i, x_k) \notin \mathcal{S}$, then we minimize the proximity value. It is computationally expensive to consider all the pairs that not in $\mathcal{S}$, which requires to compute all the headlines other than h_i when fixing x_j, or all the documents other than x_j when fixing h_i. To address this problem, we adopt the approach of negative sampling [11], which samples multiple negative pairs according to the original pairs in $\mathcal{S}$. Specifically, we aim to minimizing the following log likelihood function for each pair $(h_i, x_i) \in \mathcal{S}$:

$$\mathcal{L}_{D_P} = -\log\sigma(\mathbf{s}^{(i)}\mathbf{Q}\mathbf{z}^{(i)}) - \sum_{k=1}^{K}\mathbb{E}_{x_k\sim P_n(x)}[\log\sigma(-\mathbf{s}^{(i)}\mathbf{Q}\mathbf{z}^{(k)})] \tag{19}$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function, and $P_n(x)$ denotes the set of negative samples of documents that satisfy $(h_i, x_k) \notin \mathcal{S}$. The first term models the aligned pair of a headline and a document, the second term models the negative pairs drawn from the original pair distributions, and K is the number of negative pairs.

C. The proposed Framework: SHG

The parameters of all components (A, G, D_S, D_T and D_P) of this model are learned jointly in an end-to-end fashion. Therefore, the objective function is a minimax game among the headline generator and discriminators, shown as follows:

$$\min_{\{\theta_G,\theta_{D_S},\theta_{D_P}\}} \max_{\theta_{D_T}} \mathcal{L}_G + \alpha\mathcal{L}_{D_S} - \beta\mathcal{L}_{D_T} + \eta\mathcal{L}_{D_P} \tag{20}$$

where α, β, and η are positive hyper-parameters balancing the importance among different losses.

D. Optimization

The optimizing process is illustrated in Algorithm 1. We denote $\mathcal{S}_0 = \{(x_i, h_i)|y_i^L = 0\}$ and $\mathcal{S}_1 = \{(x_i, h_i)|y_i^L = 1\}$ for easy explanation. First, we compute the latent content representations of all documents $\{\mathbf{z}\}_{i=1}^m$ in Line 1. We then sample a mini-batch of k samples from training data. Next, we train the Professor-Forcing network from Line 5 to Line 6 and obtain the hidden states vectors $\mathbf{s}_N$ and $\tilde{\mathbf{s}}_N$. Then we train the discriminator D_T and update parameters θ_{D_T} in Line 8. At last, we train the generator G, and discriminator D_S and D_P in Line 9.

Algorithm 1 The Learning Process of **SHG**

```
Input: The two set of training tuple $\mathcal{S}_0, \mathcal{S}_1$, $\alpha$, $\beta$, $\lambda$, $\eta$,
    temperature $\gamma$, batch size $k$.
Output: The headline generator $G$ conditioned on $(\mathbf{z}, \mathbf{y}^{\mathbf{U}})$,
    generated headlines $\mathcal{O}$
1: Pre-train the document generator $A$ to obtain the content
   representations $\{\mathbf{z}_i\}_{i=1}^m = \{A(x_i)\}_{i=1}^m$
2: repeat
3:   for $l = 0, 1$ do
4:     Sample a mini-batch of $k$ samples $\{(x^i, h^i)\}_{i=1}^k$ from
       the training tuples $\mathcal{S}_l$
5:     Unroll $G$ from initial state $(\mathbf{z}_i, \mathbf{y}^L)$ by feeding $\mathbf{s}_i$,
       and get the last hidden state $\mathbf{s}_N$
6:     Unroll $G$ from initial state $(\mathbf{z}_i, \mathbf{y}^U)$ by feeding $\tilde{\mathbf{s}}_i$,
       and get the last hidden state $\tilde{\mathbf{s}}_N$
7:   end for
8:   Training the discriminator $D_T$ by gradient descent using
     Eqn 17
9:   Training the headline generator $G$ through Eqn 9, and
     the style discriminator $D_S$ through Eqn 14, and the
     discriminator $D_P$ via Eqn 19.
10: until Convergence
```

IV. Experiments

In this section, we conduct experiments on real-world datasets to demonstrate the effectiveness of the proposed framework for headline generation. Specifically, we aim to answer the following evaluation questions:

EQ1 *Consistency*: are generated clickbaits/non-clickbaits consistent with the original datasets?

EQ2 *Readability*: are generated headlines readable or not?

EQ3 *Similarity*: are generated headlines semantically similar to original documents?

EQ4 *Differentiability*: are generated clickbaits/non-clickbaits differentiable?

EQ5 *Accuracy*: can generated clickbaits/non-clickbaits help improve the detection performance?

We aim to assess the quality of generated headlines by answering **EQ1**, **EQ2** and **EQ3**, and evaluate the utility of them by answering **EQ4** and **EQ5**.

A. Datasets

TABLE I: The statistics and descriptions of the datasets

Dataset	Source	# Clickbaits	# Non-clickbaits
P	**P**rofessional Writers	5,000	16,933
M	Social **M**edia Users	4,883	16,150

Existing clickbait datasets are mainly collected from two types of sources: (1) *Professional writers* (P for short) such as news reporters or editors who come up with clickbaits for the news pieces they publish. In [4], the authors curated clickbaits from several news websites such as BuzzFeed[2], Upworthy[3], etc. The labels are annotated by human annotators and decided through majority voting. The non-clickbaits are collected from Wikinews articles [4]. In Wikinews, articles are produced by a community of contributors and each news article needs to be verified by the community before publication. The headlines need to follow a list of guidelines[5] which ensures the reliability of labeling them as non-clickbaits. We enrich the datasets in [4] by crawling the original article contents. (2) *Social media users* (M for short), who make clickbaits to lure people to click their posts. The dataset M is collected from the clickbait challenge[6]. The ground truth labels are obtained by majority voting scores voted by crowd-sourcing platforms. For each headline, the corresponding news document is also provided. In other words, each training instance is a tuple with the article document, headline, and label provided. The statistics of the two datasets are shown in Table I.

It is worth mentioning that detecting clickbaits in M is more challenging than P, because headlines in M were written by users on social media which are diverse and noisy; while headlines in P are collected from a few specific professional websites and thus the headlines with clickbaity or non-clickbaity style are more consistent and less noisy.

B. Experimental Setting

Parameter Setting. The document auto-encoder A is set as single-layer RNN with GRU cell of input/hidden dimension with $d = 64$, and the dimension of style representation is set as $l = 16$, and thus the generator G is set as a single-layer RNN with the dimension of the hidden state as 80, and the maximum length of output headline is 25. The discriminator D_T, D_P, and D_S are feed-forward networks with single hidden layer

[2]https://www.buzzfeed.com/?utm_term=.hm52wwl8E#.kv7m99nO7

[3]https://www.upworthy.com/

[4]https://en.wikinews.org/wiki/Main_Page

[5]en.wikinews.org/wiki/Wikinews:Styleguide#Headlines

[6]http://www.clickbait-challenge.org/

and a sigmoid output layer. For the document auto-encoder A, we only consider the first 100 words to as a summarization of the entire document as the input, for reducing the noise and easy training for A. The parameters α, β and η are determined through cross-validation, and are set as 1, 1, and 5.

Experimental Design. We denote the set of original headlines as $\mathcal{H}$ and the generated headlines as $\mathcal{O}$. Both $\mathcal{H}$ and $\mathcal{O}$ are divided into training sets ($\mathcal{H}_{train}$, $\mathcal{O}_{train}$) and testing ($\mathcal{H}_{test}$, $\mathcal{O}_{test}$) sets with a ratio of 3:1. To evaluate **EQ1**, we train the model on $\mathcal{H}$ and test it on $\mathcal{O}$. To evaluate **EQ2**, we utilize the widely used metric, Flesch-Kincaid readability score [12], to measure how readable of the generated headlines. For **EQ3**, we utilize Bilingual Evaluation Understudy (BLEU) [13] scores and propose a new metric uni_sim to measure the semantic similarities of the generated headlines with original documents. To assess **EQ4**, we train the predict model on $\mathcal{O}_{train}$ and test it on $\mathcal{O}_{test}$. To evaluate **EQ5**, we train on both $\mathcal{H}_{train}$ and instances from $\mathcal{O}$ and test it on $\mathcal{H}_{test}$.

Prediction Model. To ensure a fair comparison of the performance of clickbait detection with respect to the change of datasets, we fix the set of features that are used for classification among all the settings. We curate the linguistic features that are shown to be effective in previous research [14], [15], [4], resulting in a 314 dimension features, including readability score, word dictionary matching, n-grams, part-of-speech tags, and their combinations. We also apply several popular machine learning classifiers on the extracted features. Specifically, the classifiers include Logistic Regression (LogReg), Decision Tree (DTree), Random Forest (RForest), XGBoost, AdaBoost, SVM and GradBoost. We used the open-sourced *xgboost* [16] package and *scikit-learn* [17] machine learning framework in Python to implement all these algorithms. To ensure a fair comparison of features, we ran all the algorithms using default parameter settings.

Evaluation Metrics. Since our datasets are imbalanced in terms of the number of instances for clickbaits and non-clickbaits, we select the evaluation metric, Area Under the ROC Curve (AUC), to compare the prediction performance of clickbaits. We repeat all the experiments 10 times and the average results are reported. Note that for readability metric Flesch-Kincaid score [7], the higher the more difficult to read the text. For BLEU score, the higher the score is, the more similar are the pair of texts in terms of semantic meaning. We also adopt another metric to compute the semantic similarity for (headline, document) pairs based on the text embedding learned from universal sentence encoder [18]. We first obtain the embedding vectors for all documents and headlines[8], and report the average similarity scores. For example, giving the headlines $\mathcal{H} = \{h_1, h_2, ..., h_m\}$ and documents $\mathcal{X} = \{x_1, x_2, ..., x_m\}$, we define the new metric,

$$uni\_sim(\mathcal{X}, \mathcal{H}) = \frac{1}{m}\sum_{i=1}^{m} cos(uni(x_i), uni(h_i)) \tag{21}$$

where $uni(x_i)$ ($uni(h_i)$) is the universal text embedding for document x_i (headline h_i). Note that the higher the uni_sim value, the better performance that headlines preserve the content of original documents.

Baseline Methods. We compare the proposed framework with several representative text generation algorithms, SeqGAN, SVAE, and CrossA, detailed as follows:

- **SeqGAN** [19]: SeqGAN extends the GAN framework with reinforcement learning. It uses a discriminator to minimize the loss between real and fake generated texts, and a generator trained by policy gradient where the final reward signal is provided by the discriminator and the action value is determined using Monte Carlo search.
- **SVAE** [8]: SVAE uses the VAE to generate sentences, with Long Short Term Memory (LSTM) networks as both the encoder and decoder.
- **CrossA** [7]: CrossA can transfer the sentences from across different styles without parallel data. It utilizes style-specific decoders to align the style of generated sentences to the actual distribution of the style[9].

Note that SeqGAN and SVAE take the original headlines as input and generate new headlines with the same style; CrossA takes original headlines as input and generate new headlines with transferred style; while SHG takes the original headlines and documents as input and generate new headlines that aims to preserve the content of the document and transfer the headline styles. When evaluating **EQ3** for content preserving ability, we only compare CrossA and SHG because SeqGAN and SVAE are not able to transfer styles when generating headlines and thus there is no need to compare them.

C. Experimental Result: Data Quality

For evaluating the quality of generated headlines, we answer the **EQ1**, **EQ2** and **EQ3**. The experimental results are shown in Table II to Table VI, and we have the following observations:

First, in terms of data consistency (**EQ1**) between original headlines and generated headlines, we can see that the proposed framework SHG can generate headlines that are more likely to be consistent with original headlines. We observe that both CrossA and SHG have better consistency performance than other baselines. This is because they both utilize a style discriminator to ensure that the distributions of original headlines and generated headlines are close to each other through adversarial training techniques.

Next, for answering **EQ2**, we compare the readability scores in Table IV. For readability score, we can see that in general the readability of non-clickbait headlines are lower than that

[7]We apply the open source tool at: https://github.com/wimmuskee/readability-score

[8]We use the TensorFlow interface to retrieve the pre-trained embedding at: https://www.tensorflow.org/hub/modules/google/universal-sentence-encoder/1

[9]We used the off-the-shelf model with the parameter settings through: https://github.com/shentianxiao/language-style-transfer

TABLE II: Samples of Generated Headlines with Different Headline Generation Methods with Style Transfer

source document:	*apple music will give ... it will cost 10 per month ... first three months of the service for free ...*
source clickbait:	how apple s new music service compares to what s out there right now
CrossA: → nonclickbait:	how to a lot of the world s freest house is you have to be
SHG: → nonclickbait:	the economic scorecard follow apple launch music testing
source document:	*a canadian politician has denounced the recent pie thrown ... calling for the government to investigate the incident...*
source non-clickbait:	canadian politician calls for terrorism inquiry into pie throwing
CrossA: → clickbait:	this woman s adorable flute and it is actually bad
SHG: → clickbait:	here s the results from a pie throw

TABLE III: **EQ1**:The Data Consistency between Generated Headlines $\mathcal{O}$ and original headlines $\mathcal{H}$. Training data is $\mathcal{H}$ and test data is $\mathcal{O}$.

Data	Classifier	SeqGAN	SVAE	CrossA	SHG
P	LogReg	0.501	0.621	0.957	**0.963**
	DTree	0.504	0.603	**0.924**	0.920
	RForest	0.504	0.716	0.924	**0.976**
	XGBoost	0.500	0.722	0.938	**0.977**
	AdaBoost	0.500	0.714	0.957	**0.977**
	SVM	0.503	0.712	0.937	**0.963**
	GradBoost	0.501	0.722	0.933	**0.969**
M	LogReg	0.500	0.576	0.666	**0.763**
	DTree	0.53	0.623	0.687	**0.750**
	RForest	0.501	0.662	0.722	**0.825**
	XGBoost	0.501	0.610	0.660	**0.742**
	AdaBoost	0.500	0.671	0.691	**0.749**
	SVM	0.500	0.620	0.648	**0.725**
	GradBoost	0.501	0.674	0.692	**0.742**

TABLE IV: **EQ2**: The Readability score comparison of the generated headlines $\mathcal{O}$ on different generation methods.

Data	Methods	Clickbait	Non-Clickbait
P	SeqGAN	14.45	14.54
	SVAE	8.64	10.38
	CrossA	8.98	10.48
	SHG	**8.45**	**10.16**
M	SeqGAN	14.48	15.88
	SVAE	9.52	10.79
	CrossA	9.86	10.56
	SHG	**9.36**	**10.04**

of clickbait headlines on both datasets. This is because non-clickbaits are longer than clickbaits in general and thus decrease the readability performance. Moreover, among all the text generation methods, the proposed SHG can produce most readable headlines mostly, which demonstrates the ability to generate more easily readable headlines.

TABLE V: **EQ3**: The Average BLEU (BLEU-4) Score Comparison of Generated Headlines. $\mathcal{H}$ indicates original headlines, and $\mathcal{O}$ represents the generated headlines.

Data	Headlines	Methods	Clickbait	Non-Clickbait
P	$\mathcal{H}$		0.555	0.527
	$\mathcal{O}$	CrossA	0.407	0.432
		SHG	**0.453**	**0.446**
M	$\mathcal{H}$		0.541	0.534
	$\mathcal{O}$	CrossA	0.432	0.437
		SHG	**0.451**	**0.442**

Next, for the ability of preserving the contents of original documents (**EQ3**), we first compute the average BLEU-4 (BLEU score for 4-grams) (the results of BLEU-1) for each pair of headline and document as in in Table V. We can see the proposed SHG can achieve better performances than CrossA. We further compare the average uni_sim scores to illustrate the semantic similarities in the universal latent space as shown in Table VI. Similarly, we can see that: 1) The uni_sim score for the pair of original headlines and documents are higher than that of generated headlines and documents; 2) SHG performs better that CrossA for uni_sim values on both datasets. The results indicate the importance that SHG utilizes the pre-trained paragraph auto-encoder and a pair discriminator to ensure the representations of generated stylized headlines are correlated with that of documents.

TABLE VI: **EQ3**: The Average Uni_sim Value Comparison of Generated Headlines. $\mathcal{H}$ indicates original headlines, and $\mathcal{O}$ represents the generated headlines.

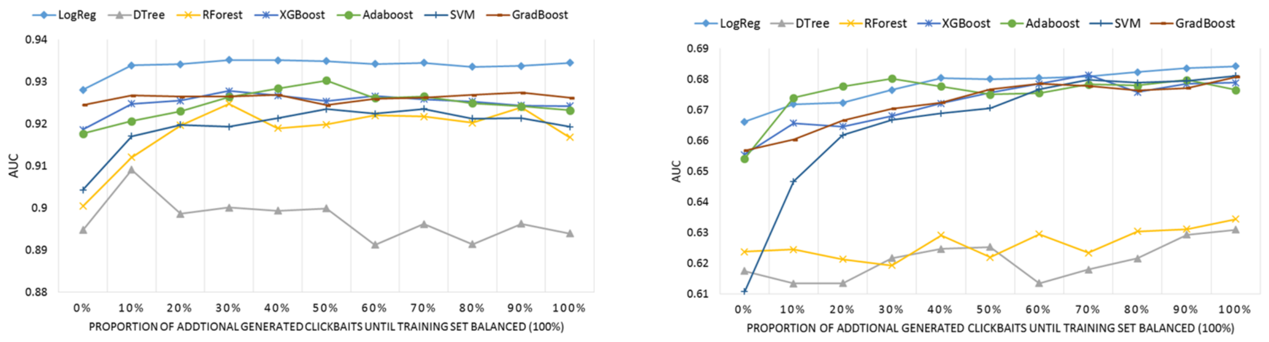
Data	Headlines	Methods	Clickbait	Non-Clickbait
P	$\mathcal{H}$		0.63	0.81
	$\mathcal{O}$	CrossA	0.20	0.22
		SHG	**0.37**	**0.40**
M	$\mathcal{H}$		0.64	0.81
	$\mathcal{O}$	CrossA	0.26	0.34
		SHG	**0.34**	**0.38**

To illustrate the qualities of generated headlines, we show several examples in Table II. We can see that SHG can generate a clickbait style headline that preserves the content “pie throw”; In addition, SHG can produce the a headline without clickbait style and keep semantic meaning from the original document to some degree. It shows the potential use of SHG to suggest a non-clickbait style headline for a given document.

D. Experimental Result: Data Utility

We further compare the utility of generated headlines. We first perform clickbait detection on generated datasets $\mathcal{O}$ to check whether they are differentiable or not. We then use the generated headlines as auxiliary data to enrich the original training dataset, and compare the utility of improving the detection performance on original datasets. The experiments are illustrated in Table VII and Table VIII and Figure 2, and we have the following observations:

For the differentiability of the generated headlines (**EQ4**), we can see that the synthetic headlines generated by SHG can consistently outperform other baselines for the clickbait



(a) P (b) M

Fig. 2: The performance improvement w.r.t. the proportion of additional generated clickbaits from SHG on AUC score.

TABLE VII: **EQ4**: The prediction performance of generated headlines $\mathcal{O}$ on AUC. The training data is $\mathcal{O}_{train}$ and test data is $\mathcal{O}_{test}$.

Data	Classifier	SeqGAN	SVAE	CrossA	SHG
P	LogReg	0.697	0.710	0.794	**0.864**
	DTree	0.701	0.766	0.791	**0.816**
	RForest	0.685	0.794	0.797	**0.849**
	XGBoost	0.701	0.795	0.795	**0.848**
	AdaBoost	0.694	0.800	0.793	**0.848**
	SVM	0.646	0.795	0.787	**0.847**
	GradBoost	0.702	0.797	0.792	**0.848**
M	LogReg	0.625	0.663	0.744	**0.855**
	DTree	0.612	0.712	0.771	**0.934**
	RForest	0.598	0.724	0.783	**0.883**
	XGBoost	0.616	0.651	0.697	**0.872**
	AdaBoost	0.642	0.667	0.708	**0.872**
	SVM	0.510	0.590	0.693	**0.890**
	GradBoost	0.624	0.654	0.708	**0.885**

detection performance in terms of AUC score w.r.t different classifiers on both datasets. It shows that: i) our generated headlines have more clear style differences between clickbait and non-clickbait styles; ii) headlines with the same style demonstrate highly consistent features. It demonstrates the robustness of style-transferring of the proposed method.

In addition, we compare the performance improvement by adding generated headlines as additional training data in original datasets (**EQ5**). Since clickbait headlines are usually sparse, we add different proportion of the generated clickbaits by SHG until the training set is balanced (see Figure 2). We can see that with the increase of newly generated clickbaits, the AUC scores does not necessarily increase w.r.t. different classifiers on both datasets. For example, in P dataset, the predict performance may even decrease for DTree classifier with more generated dataset added. In a practical setting, we can actually add 10% or 20% of newly generated clickbaits, which already has good performance improvement in most cases.

Moreover, we also add different proportion of generated clickbaits from other baseline methods, and select the best performances for comparison (see Table VIII). We can see that: i) the incorporation of synthetic headlines which are generated by SeqGAN decrease the performance comparing with original datasets, which is because the added generated headlines add additional noise to the training datasets; and ii) The headlines generated by SVAE, CrossA, and SHG can increase the performance of clickbait detection to some extent; iii) SHG can consistently outperform SVAE and CrossA on the performance improvement, which demonstrates that SHG can generate headlines that help enrich original sparse datasets and build a more powerful clickbait detection algorithm.

We can see that in general the detection performance on P is higher than M. This is because the sources of headlines in M are more diverse and noisy than P, and the classification boundaries are more clear in P. Thus, the styles are easier to be transfered for P due to the consistency and less noise within the instances of the same style.

To sum up, we conclude from the experiments that (1) the proposed framework can generate high quality stylized headlines from documents which have better performance w.r.t consistency, readability, and semantic similarity; (2) The headlines generated from SHR are robust in distinguishing between clickbait and non-clickbait styles, and can significantly help improve clickbait detection by enriching training data.

V. Related Work

In this section, we review the related work from following perspectives: i) Headline generation; ii) Style transfer; iii) Adversarial training on discrete variables; and iv) Clickbait detection.

A. *Headline Generation*

Headline generation is to construct a headline-style abstracts from a single document, which aims to produce informative content describing the salient theme or event in the news article [20]. Extractive methods compress the document sentences and select a subset of actual sentences from the original documents to produce the headline [20], [21]. Most extractive methods focus on selecting a subset of actual sentences and may not be able to generate coherent and compact summary. Generative methods mainly adopt a encoder-decoder framework, where encoder reads the article and encodes it to a sequence of latent representations [22]; the decoder outputs

TABLE VIII: **EQ5**: The performance improvement comparison of original headlines $\mathcal{H}$ on AUC. The training data consists of $\mathcal{H}_{train}$ and O, and test data is $\mathcal{H}_{test}$. Org shows the performance in original dataset, i.e., training data is $\mathcal{H}_{train}$ and testing data is $\mathcal{H}_{test}$. The numbers in brackets show the relative improvements compare with original dataset.

Data	Classifier	Org	SeqGAN	SVAE	CrossA	SHG
P	LogReg	0.928	0.900 ($\downarrow$ 3.02%)	0.933 ($\uparrow$ 0.54%)	0.932 ($\uparrow$ 0.64%)	**0.936 ($\uparrow$ 0.86%)**
	DTree	0.894	0.882 ($\downarrow$ 1.34%)	0.908 ($\uparrow$ 1.57%)	0.900 ($\uparrow$ 0.67%)	**0.910 ($\uparrow$ 1.79%)**
	RForest	0.900	0.893 ($\downarrow$ 0.78%)	0.912 ($\uparrow$ 1.33%)	0.916 ($\uparrow$ 1.78%)	**0.925 ($\uparrow$ 2.78%)**
	XGBoost	0.919	0.914 ($\downarrow$ 0.54%)	0.923 ($\uparrow$ 0.43%)	0.926 ($\uparrow$ 0.76%)	**0.928 ($\uparrow$ 0.98%)**
	AdaBoost	0.917	0.896 ($\downarrow$ 2.29%)	0.921 ($\uparrow$ 0.44%)	0.921 ($\uparrow$ 0.44%)	**0.931 ($\uparrow$ 1.64%)**
	SVM	0.904	0.898 ($\downarrow$ 0.66%)	0.917 ($\uparrow$ 1.44%)	0.920 ($\uparrow$ 1.77%)	**0.923 ($\uparrow$ 2.10%)**
	GradBoost	0.921	0.914 ($\downarrow$ 0.76%)	0.924 ($\uparrow$ 0.33%)	0.926 ($\uparrow$ 0.54%)	**0.928 ($\uparrow$ 0.76%)**
M	LogReg	0.667	0.614 ($\downarrow$ 7.95%)	0.680 ($\uparrow$ 1.95%)	**0.685 ($\uparrow$ 2.70%)**	0.684 ($\uparrow$ 2.55%)
	DTree	0.618	0.612 ($\downarrow$ 0.97%)	0.620 ($\uparrow$ 0.32%)	0.622 ($\uparrow$ 0.65%)	**0.632 ($\uparrow$ 2.27%)**
	RForest	0.623	0.610 ($\downarrow$ 2.09%)	0.634 ($\uparrow$ 1.77%)	0.627 ($\uparrow$ 0.64%)	**0.634 ($\uparrow$ 3.93%)**
	XGBoost	0.655	0.643 ($\downarrow$ 1.84%)	0.668 ($\uparrow$ 1.98%)	0.671 ($\uparrow$ 2.44%)	**0.681 ($\uparrow$ 3.97%)**
	AdaBoost	0.654	0.639 ($\downarrow$ 2.29%)	0.664 ($\uparrow$ 0.65%)	0.671 ($\uparrow$ 2.60%)	**0.680 ($\uparrow$ 3.98%)**
	SVM	0.618	0.611 ($\downarrow$ 1.13%)	0.651 ($\uparrow$ 5.34%)	0.660 ($\uparrow$ 6.80%)	**0.681 ($\uparrow$ 10.19%)**
	GradBoost	0.657	0.643 ($\downarrow$ 2.13%)	0.667 ($\uparrow$ 1.52%)	0.671 ($\uparrow$ 2.13%)	**0.682 ($\uparrow$ 3.81%)**

a summary word-by-word using the latent representations. In [23], it propose an attention-based encoder-decoder framework to generate headlines. In addition, [24] incorporate the position information of words into a RNN encoder and shows to be effective. [25] tries different mechanisms to reduce vocabulary size and capture relevant keywords. These methods generate sentences without incorporating styles to them.

B. Style Transfer

Style transfer has been widely studied in computer vision in recent years [26]. Inspired by the power of CNN, [26] first studied to mitigate the semantic content of one image and combine different styles from other images to reproduce images with famous painting styles. Due to the discrete property of text data and unavailable of parallel data, previous methods on vision domain cannot be directly applied to text domain. Recently, several papers are proposed for style transfer with non-parallel data [7], [6]. In [27], they propose to guide the latent representation learning by using a classifier. In [6], they use the VAE to encode sentence to content representation which is disentangled from source style, and then recombine it with target style to produce its counterpart. Similarly, [28] use VAE to encode sentences to both content and style representations across two domains simultaneously using adversarial networks.

C. Adversarial training on discrete variables

Recently there are increasing works focusing on adversarial training over discrete sequence data [19], [29], [30]. In [29], they propose providing the discriminator with the intermediate hidden state vectors rather than its sequence outputs, which makes the system differentiable for back propagation training. SeqGAN [19] applies GAN [10] to discrete sequence generation by directly optimizing the discriminator's rewards using policy gradient reinforcement learning. Other approaches use continuous approximation to represent discrete tokens to facilitate the gradient propagation process [31], [32], [6]. Continuous approximation use the Gumbel-softmax function [33] to transform the one-hot vector represented sampled tokens into a probabilistic vector that is differentiable for training. We use the Professor-Forcing structure with continuous approximation for adversarial training in this paper.

D. Clickbait Detection

Potthast *et al.* proposed a machine learning approach to detect clickbait Tweets by using features extracted from teaser messages, linked web pages and tweet meta information [34]. Blom *et al.* analyzed the how clickbaits use two forms of forward referencing, namely discourse deixis and cataphora, to lure readers to click target links [35]. Recently Chakraborty *et al.* attempted to automatically detect clickbaits using various linguistic features and deployed a Chrome extension to warn readers about clickbaits. Lal *et al.* [5] introduce a model stacking RNN, attention mechanism, and image embeddings to detect clickbaits. Rony *et al.* [2] use distributed word embeddings, extending Skip-gram model, trained from a Facebook corpus and learn vector representation of headlines to feed into a RNN-LSTM architecture model.

VI. Conclusion and Future Work

Building effective clickbait detection models often suffers from the shortage of labeled training data. In this paper, addressing this bottleneck of supervised learning workflow, we propose a headline generation framework, named as SHG, to generate stylized headlines from original documents, and demonstrate its utility in improving the accuracy of clickbait detection problem. The proposed framework highlights two major dimensions: i) a generator learning dimension, which extracts content representations and incorporate style representations to generate headlines; ii) a discriminator learning dimension, which guides the generating process through constraints by enforcing pair correspondences, style differences, and transfer maintenances. Experiments on real-world datasets demonstrate the effectiveness of the proposed SHG framework for generating high quality and useful headlines.

There are several interesting directions that need further investigations. First, in this paper, we only considered the style from the perspective of clickbaits and/or non-clickbaits. In addition, we can also explore the generalization capacity of SHG on other styles such as positive-negative sentiment style and academic-news reporting style [36]. Second, we currently focus on generating short texts, i.e., headlines. We believe generating longer sentences/paragraphs [37] remains an open problem and is critical for many potential applications such as fake news detection [1] and fraud review detection [38]. Third, a detailed theoretical and practical analysis of SHG, such as the ability to learn disentangled representations of content and style, is also worth investigating.

VII. Ackowledgments

This material is based upon work supported by, or in part by, the National Science Foundation (NSF) under grant #1614576 and Office of Naval Research (ONR) under grant N00014-17-1-2605. Thai Le and Dongwon Lee are supported by NSF awards #1422215, #1663343, #1742702, and #1820609.

References

[1] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[2] M. M. U. Rony, N. Hassan, and M. Yousuf, "Diving deep into clickbaits: Who use them to what extents in which topics with what effects?" in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. ACM, 2017, pp. 232–239.

[3] J. Avery, M. Almeshekah, and E. Spafford, "Offensive deception in computing," in *International Conference on Cyber Warfare and Security*. Academic Conferences International Limited, 2017, p. 23.

[4] A. Chakraborty, B. Paranjape, S. Kakarla, and N. Ganguly, "Stop clickbait: Detecting and preventing clickbaits in online news media," in *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*. IEEE, 2016, pp. 9–16.

[5] S. Gairola, Y. K. Lal, V. Kumar, and D. Khattar, "A neural clickbait detection engine," *arXiv preprint arXiv:1710.01507*, 2017.

[6] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *International Conference on Machine Learning*, 2017, pp. 1587–1596.

[7] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola, "Style transfer from non-parallel text by cross-alignment," *arXiv preprint arXiv:1705.09655*, 2017.

[8] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[11] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[12] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, and B. S. Chissom, "Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel," Naval Technical Training Command Millington TN Research Branch, Tech. Rep., 1975.

[13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

[14] P. Biyani, K. Tsioutsiouliklis, and J. Blackmer, "" 8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality." in *AAAI*, 2016, pp. 94–100.

[15] X. Cao, T. Le *et al.*, "Machine learning based detection of clickbait posts in social media," *arXiv preprint arXiv:1710.01977*, 2017.

[16] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[18] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar *et al.*, "Universal sentence encoder," *arXiv preprint arXiv:1803.11175*, 2018.

[19] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in *AAAI*, 2017, pp. 2852–2858.

[20] B. Dorr, D. Zajic, and R. Schwartz, "Hedge trimmer: A parse-and-trim approach to headline generation," in *Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5*. Association for Computational Linguistics, 2003, pp. 1–8.

[21] E. Alfonseca, D. Pighin, and G. Garrido, "Heady: News headline abstraction through event pattern clustering." in *ACL (1)*, 2013, pp. 1243–1253.

[22] S. Wang and H. Liu, "Deep learning for feature representation," *Feature Engineering for Machine Learning and Data Analytics*, p. 279, 2018.

[23] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *arXiv preprint arXiv:1509.00685*, 2015.

[24] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in *NAACL*, 2016, pp. 93–98.

[25] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.

[26] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

[27] J. Mueller, D. Gifford, and T. Jaakkola, "Sequence to better sequence: continuous revision of combinatorial structures," in *International Conference on Machine Learning*, 2017, pp. 2536–2544.

[28] S.-Q. Shen, Y.-K. Lin, C.-C. Tu, Y. Zhao, Z.-Y. Liu, M.-S. Sun *et al.*, "Recent advances on neural headline generation," *Journal of Computer Science and Technology*, vol. 32, no. 4, pp. 768–784, 2017.

[29] A. M. Lamb, A. G. A. P. GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Advances In Neural Information Processing Systems*, 2016, pp. 4601–4609.

[30] Y. Li, Q. Pan, S. Wang, T. Yang, and E. Cambria, "A generative model for category text generation," *Information Sciences*, vol. 450, pp. 301–315, 2018.

[31] Y. Zhang, Z. Gan, and L. Carin, "Generating text via adversarial training," in *NIPS workshop on Adversarial Training*, 2016.

[32] M. J. Kusner and J. M. Hernández-Lobato, "Gans for sequences of discrete elements with the gumbel-softmax distribution," *arXiv preprint arXiv:1611.04051*, 2016.

[33] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[34] M. Potthast, S. Köpsel, B. Stein, and M. Hagen, "Clickbait detection," in *European Conference on Information Retrieval*. Springer, 2016, pp. 810–817.

[35] J. N. Blom and K. R. Hansen, "Click bait: Forward-reference as lure in online news headlines," *Journal of Pragmatics*, vol. 76, pp. 87–100, 2015.

[36] Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan, "Style transfer in text: Exploration and evaluation," *arXiv preprint arXiv:1711.06861*, 2017.

[37] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," *arXiv preprint arXiv:1709.08624*, 2017.

[38] Y. Yao, B. Viswanath, J. Cryan, H. Zheng, and B. Y. Zhao, "Automated crowdturfing attacks and defenses in online review systems," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1143–1158.