Generating Topic-Preserving Synthetic News

Ahmadreza Mosallanezhad
Computer Science and Engineering
Arizona State University
amosalla@asu.edu

Kai Shu
Department of Computer Science
Illinois Institute of Technology
kshu@iit.edu

Huan Liu
Computer Science and Engineering
Arizona State University
huanliu@asu.edu

***Abstract*—The text generation methods have witnessed great success in text summarization, machine translation, and synthetic news generation. However, these techniques may be abused to generate disinformation and fake news. To better understand the potential threats of synthetic news, we develop a novel generation method RLTG to generate topic-preserving news content. The majority of existing text generation methods are either controlled by specific attributes or lack topic consistency between the input claims and output news, making synthetic news less coherent and realistic. In this paper, we study the problem of topic-preserving synthetic news generation by proposing a novel deep reinforcement learning-based method to control the output of large pre-trained language models. Experiment results on real-world datasets demonstrate that the news contents generated by RLTG are topic-consistent and realistic.**

***Index Terms*—Text Generation, Reinforcement Learning, Adversarial Training**

I. Introduction

Text generation is an important task for Natural Language Processing (NLP). With the rise of deep neural networks such as Recurrent Neural Networks (RNNs) and Long Shot Term Memory (LSTM) cells [1], there has been significant performance improvement in language modeling and text generation. Text generation has many different applications such as paraphrase generation and data augmentation. One important application of text generation in NLP is synthetic news content generation [2].

Recently, internet has proliferated a plethora of disinformation and fake news [3], [4]. Moreover, recent advancements in language models such as GPT-2 [5] allow one to generate synthetic news based on limited information. For example, models like Generative Adversarial Network (GAN) [6] can generate long readable text from noise, and GPT-2 [5] can write news stories and fiction stories given simple contexts such as part of a sentence or a topic. In the context of news generation, Grover is a causal language model that can generate fake news using different variables such as *domain*, *date*, *authors*, and *headline* [2]. While Grover is shown effective, it requires many conditional variables to generate relevant news. To study machine-generated news, we propose a model to generate realistic synthetic news. Throughout this paper, we refer to realistic news as news similar to human-written words. The crucial task of synthetic news generation enables us to (1) automatically generate news and (2) use the synthetic news to study the differences between human-generated and machine-generated news. For example, one major problem in fake news detection is the challenge to differentiate between human and machine-generated text [2].

With the advances in language models (e.g., GPT-2 and GPT-3), miscreants can leverage them to spread fake news through social media. To tackle this problem, as the first step, we need ample synthetic news with which researchers can study the nuances between human- and machine-generated text to detect disinformation on social media.

Existing methods may fall short when generating realistic news controlled by a specific context. For instance, fake news usually has a catchy style and should stay on topic to make its audience believe it, as in the example of "A shocking news report claims Kourtney Kardashian's pregnant again". The shortcomings in existing language models and the lack of a proper machine-generated news dataset underscore the importance of topic-preserving and stylized synthetic news generation. Moreover, fine-tuning language models does not help us in this matter as it is non-trivial to enforce topic-preservation on a language model directly. In essence, we address the challenge of generating topic-preserving realistic synthetic news.

Our solutions to these challenges result in a novel framework RLTG (Reinforcement Learning-based Text Generator), for generating topic-preserving realistic news. The proposed framework RLTG consists of three major components: (1) a language model component to generate a probability distribution over a vocabulary for the next word, given a text input; (2) a Reinforcement Learning (RL) component capable of leveraging the language model to control news generation; and (3) a fake news detection module as an adversary to help the RL agent generate realistic fake news contents. This work can be used as a stepping stone to further study the differences between human- and machine-generated news, and improving fake news detection methods. Our contributions are summarized as follows:

- We study a novel problem of topic-preserving and realistic synthetic news content generation.
- We propose a principled framework, RLTG, which uses language model and deep reinforcement learning along with adversary regularization to generate realistic synthetic news contents.
- We conduct experiments on real-world datasets using quantitative and qualitative metrics to demonstrate the effectiveness of RLTG for synthetic news generation.

II. Related Work

In this section, we briefly describe the related work on (1) neural news generation; (2) adversarial training; and (3) reinforcement learning for text generation:

A. *Neural news generation*

Text generation is a crucial task in NLP and is used in its different applications [7], [8]. Many early methods for text generation use different techniques to train Generative Adversarial Networks (GAN). As GANs cannot be used for text generation due to the discrete nature of the problem, the early work try to solve the problem of back propagation for updating the generator. Several methods [6], [9], [10] have been proposed to alleviate this problem. MaskGAN [10] tries to generate text using both GAN and actor-critic networks. Another method from [9] uses a new training method for the discriminator in a GAN to learn a better text generator against the discriminator. Finally, in LeakGAN [6], unlike other GAN-based methods that the discriminator and generator are trained against each other, it uses the discriminator to help the generator predict the next word.

To solve the problem of controllable text generation for a specific topic, Hu et al. propose a method for controlling the text generation process using different classifiers as discriminators in a GAN [11]. Another recent work by Dathathri et al. proposed PPLM that uses gradient difference on a pre-trained language model to control its output toward a given attribute. They further use different classifiers to calculate the gradient difference [12]. Moreover, Hu et al. proposed to use several discriminators in a GAN to control the text generation process [11]. What makes our work different than these methods is that in our work we focus on controlling the output toward a given topic, instead of a given class attribute. Lastly, Zellers et al. [2] use transformers to build a causal language model in order to generate text given a desired article's parameters.

B. *Adversarial training on discrete variables*

Recently, interesting works has been done that use adversarial training over discrete sequence data [13]–[17]. Lamb et al.propose providing the discriminator with the intermediate hidden state vectors rather than its sequence outputs, which makes the loss function differentiable for back propagation training [16]. Due to the problems in generating text with Recurrent Neural Netowrks such as quality deterioration in long sentences, Bengio et al. propose a method to overcome this problem by using scheduled sampling. They call the quality deterioration as exposure bias in RNNs [13]. However, Huszar et al. showed that the scheduled sampling method is inconsistent and can result in an unstable results [14].

Yu et al. propose a novel method named SeqGAN. They apply a Generative Adversarial Network [18] to discrete sequence generation by directly optimizing the discriminator's rewards using policy gradient reinforcement learning [15]. Other approaches use continuous approximation to represent discrete tokens to facilitate the gradient propagation process [11], [19]. Continuous approximation uses the Gumbel-softmax function [20] to transform the one-hot vector into a probabilistic vector that is differentiable for training.

Efforts have been made to generate diverse and high-quality text [6], [21]. Guo et al. propose a new method for generating long text using adversarial training. They leverage the hidden states of an adversary as leaked information in order to optimize a GAN to generate long text [6]. To broaden the domains of generated text Wang et al. propose a method which uses a multi-class classifier as a discriminator. It further uses multiple generators alongside the discriminator to optimize the model [22]. Moreover, Zhang et al. propose a novel method, TextGAN, to alleviate the problems of generating text using GAN. They use LSTM as a generator, and a Convolutional Neural Network as a discriminator [21].

C. *Reinforcement learning in text generation*

In the past years, reinforcement learning has shown to be useful in improving model parameters [23], [24]. Furthermore, it can be used as a standalone algorithm for different purposes such as dialog or paraphrase generation. Fedus et al. propose a method for overcoming the problems of generating text via GAN. They use reinforcement learning to tune parameters for an LSTM based generator [10]. Zichao Li et al. propose a method for generating paraphrase using inverse reinforcement learning. They use an RL setting to tune a generator's parameter toward generating paraphrases [24]. Another inspiring work by Jwei Li et al. shows using reinforcement learning we can build an agent capable of engaging in a two-person dialog [23]. To generate diverse text, Shi et al. propose a method which uses a generator in an RL setting. The difference between their work and other similar work is that they also change the parameters of the reward function during the training process [25]. Fan et al. [26] propose using inverse RL to solve the problem of mode collapse in GAN, meaning that during the training of GAN, the discriminator becomes too powerful that we cannot train a generator against it. Another work [27] models the problem of GAN text generation using RL. They use a reward function as a feedback to the generator. Although these methods can be used to generate text, they cannot be used to generate text for a specific domain.

Inspired by these methods, we study the novel problem of topic-preserving synthetic news generation using RL. Earlier research uses RL to update a model's parameters, but in this work, we focus on using RL *alongside* a language model to control its output towards news generation. We use this novel model to train an agent that uses a language model's output to control the generation process according to a reward function that considers topic similarity and other quality control factors.

III. Proposed Model - RLTG

In this section, we discuss the adversarial reinforcement learning-based synthetic news content generator. The input of this model is a topic $\mathbf{S_0} = \{w_1^s, w_2^s, ..., w_k^s\}$. Our model, then, generates a new sequence $\mathbf{S_T} = \{w_1, ..., w_k, w_{k+1}, ..., w_T\}$ which is the generated news content.

In this paper our goal is to generate synthetic news content $\mathbf{S_T}$ given topic $\mathbf{S_0}$. The generated news content $\mathbf{S_T}$ should be related to the given topic $\mathbf{S_0}$ and it should have a similar style to real news. A piece of news content has a similar style to real news if it cannot be detected as fake news using a classifier. Here, we study the following problem:

Problem Statement: Let $\mathcal{X} = \{(\mathbf{S_0^1}, \mathbf{x_1}), (\mathbf{S_0^2}, \mathbf{x_2}), ..., (\mathbf{S_0^N}, \mathbf{x_N})\}$ denote a set of N news with topic $\mathbf{S_0}$ and content $\mathbf{x}$. Both topic $\mathbf{S_0} = \{w_0^s, w_1^s, ..., w_k^s\}$ and news content $\mathbf{x} = \{w_0, w_1, ..., w_l\}$ consist of words w. We consider topic as the news title or the first few words of a news content. In general, S_t shows the generated text at time t. Given a set of news dataset $\mathcal{X}$, learn a reinforcement learning agent F that can generate news content $\mathbf{S_T}$ based on a given topic $\mathbf{S_0}$ such that: (1) $\mathbf{S_T}$ is related to the given topic $\mathbf{S_0}$; and (2) $\mathbf{S_T}$ has a similar style to real news.

Our model consists of several components: (1) a language model component that is in charge of generating a probability distribution over vocabulary words ; (2) an RL component that will select a word based on the language model's output; and (3) an adversarial component that will help the RL agent choose proper words from the language model's output. First, we go through news content generation using adversarial RL, and then we discuss using an adversary to generate realistic fake news.

A. Topic-preserving News Generation

Existing language models are proposed to generate general or domain-specific texts [5], [28]. Although we can fine-tune these models (e.g., fine-tuning GPT-2) according to our need using a related dataset, we do not have control over its output because we cannot enforce topic-preservation or realistic synthetic news generation on the model. Following the success of Reinforcement Learning (RL) [25], we propose an adversarial RL method to control the generated output of a language model. In recent studies, RL has been used to update a model's parameters [23], [24]. In this work we explore a new direction by using RL as a standalone component to leverage the language model's output to generate text. The main advantage of using RL alongside GPT-2 is that we can use non-differential metrics in the reward function to generate a coherent text. Moreover, it enables us to have more control on the output of the language model by leveraging adversaries or changing the reward function.

In adversarial RL an agent keeps interacting with a defined environment to learn an optimized action selection policy $\pi(s)$ for each state. An RL agent is trained to choose the next word w for current generated news $\mathbf{S_t}$ according to a reward function and an adversary.

Figure 1 shows the high-level structure of RLTG. In this model, the adversarial reinforcement learning agent gets a state s_t as input, then returns an action a_t which indicates an index to one of the top words from the language model L's output. Each interaction between the agent and the environment creates an experience tuple $(s_t, a_t, s_{t+1}, r_{t+1})$, meaning that the agent chose action a_t given the state s_t. After action a_t, the state will change to s_{t+1} and the environment returns reward r_{t+1}. This tuple is then used to train the agent. An RL model relies on four main parts: environment, state, action, and reward function.

- ***Environment*** is where the RL agent interacts with to learn the best action for each state. In our problem, the environment includes a language model L, an adversary ADV, and a state creator component M. The language model L takes an input text and returns a probability distribution over vocabulary $P \in \mathbb{R}^{1\times|V|}$ and hidden states $H \in \mathbb{R}^{1\times e}$, where e indicates the embedding size. The adversary ADV gets an input text and returns a score for the reward function. Finally, the state creator M gets the outputs of the language model as input and returns a vector $\mathbf{s} \in \mathbb{R}^{1\times|s|}$ ($|s|$ shows state size) which acts as the input state s for the agent.

- ***State*** shows the agent's current situation. The agent uses the state to determine a subsequent optimal action. The state is the output of the state creator component M. Because our goal is to select the best next word for the current generated news $\mathbf{S_t}$ at time t, the state should contain information about both the context of the current generated news $\mathbf{S_t}$, and information about the next word choices. To this end, we design two separate neural networks AE^1 and AE^2 to encode this information. AE^1 is used to create the context vector c_g using hidden state $\mathbf{H}$ from the language model L's output, while the AE^2 is used to create a context vector c_w given previous top K words of the language model L's output. For both cases, we train and use autoencoders [29]. An autoencoder is an unsupervised neural network which learns to compress and encode data and then to reconstruct the input using the encoded data. It has two components, an encoder and a decoder. The encoder takes an input and returns a vector $\mathbf{v}$ which is interpreted as the context vector, containing important information about the input. The decoder is the reverse of the encoder: given the encoded vector $\mathbf{v}$, it tries to reconstruct the original input to the network. After training an autoencoder, we can use the context vector v containing important information about the input [29].

In our method, the first autoencoder AE^1, gets the hidden state $\mathbf{H}$ as input and returns the reconstructed hidden state $\mathbf{H'}$. This autoencoder uses Multi Layer Perceptron (MLP) networks as both encoder and decoder. The purpose of this autoencoder is to reduce dimension of the hidden state $\mathbf{H}$. After training this autoencoder on a set of hidden states $\mathbf{H}$, we get the output of the encoder as context vector $\mathbf{c_g}$.

The second autoencoder AE^2, inspired by [30], uses Convolutional Neural Network (CNN) as both encoder and decoder. To this end, each word from top K words is passed through an embedding layer to convert it to a vector $\mathbf{w} \in \mathbb{R}^{1\times e}$. The embedded words $\mathbf{w}$ are then concatenated to form a matrix m with size of $(K \times e)$. After training this autoencoder using different top K words, we consider the output of the encoder as the context vector $\mathbf{c_w}$. Having both context vectors c_g and c_w,

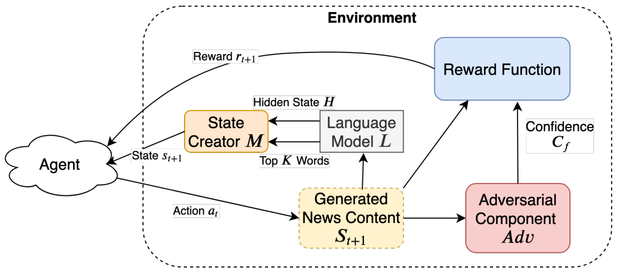


Fig. 1: The proposed model architecture. The environment consists of a language model L, reward function, state creator M, and adversary Adv. The agent keeps interacting with the environment to find the best action a for each given state s.

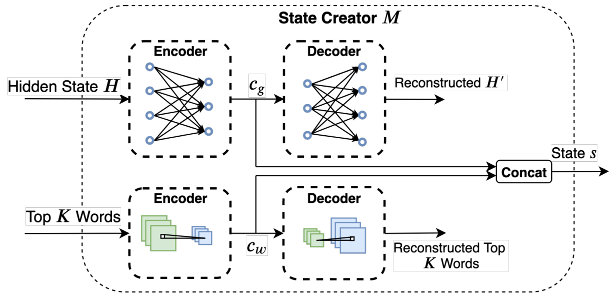


Fig. 2: The architecture of state creator M. It leverages two autoencoders to create state s for the RL agent.

we then concatenate both context vectors $\mathbf{s} = Concat(\mathbf{c_g}, \mathbf{c_w})$ to create the state for the RL environment.

• ***Actions*** indicate the agent's response to a given state s. As the agent's goal is to select words, the action set A can be equal to choosing a word from the vocabulary set $\mathcal{V}$. By choosing $\mathcal{V}$ as the agent's action set, we encounter two problems: First, it takes a long time to train an agent on a large action set as the agent should try every action to find the best action a for each given state s [31]. Secondly, by having a large action set A, the agent may not be able to see every state-action set (s, a) in a limited time, and, it may result in underfitting [31]. To solve these problems, we make use of the language model L's output. One of the outputs of the language model is the probability distribution over vocabulary $\mathcal{V}$. The probability distribution indicates what are the best options to sample the next word for a given text $\mathbf{S_t}$. In this paper, we select top K words of the probability distribution as the action set, leading to a small action set.

• ***Reward Function*** evaluates agent's actions for each given set (s_t, a_t). During training, the agent uses the reward function to learn the best strategy for selecting actions. In this paper, the goal is to generate a synthetic news content which is related to a given topic. To this end, we use cosine similarity to measure similarity between the given embedded topic $\mathbf{S_0}$ and the current generated synthetic news $\mathbf{S_t}$. The reason behind using the embedded topic and generated synthetic news at time t, is that using the exact words in the Cosine similarity function may result in an agent that chooses topic word to maximize this similarity:

$$CosineSim(\mathbf{S'_0}, \mathbf{S'_t}) = \frac{\mathbf{S'_0} \cdot \mathbf{S'_t}}{||\mathbf{S'_0}|| \cdot ||\mathbf{S'_t}||} \tag{1}$$

where S' is the embedded topic/news using the language model L. We use the language model L's hidden state H as the embedding for an input text as it shows the context of an input text [32].

Furthermore, for generating news content, the model should consider the writing style of news content. In this paper, we define style as having a similar word sequence as the reference news. To this end, for a given synthetic generated news $\mathbf{S_t}$, we calculate the BLEU score [33] between $\mathbf{S_t}$ and news contents $\mathcal{X}$ to maintain news style. The BLEU score simply measures how many words overlap between the generated news $\mathbf{S_t}$ and the reference news contents $\mathcal{X}$. As the BLEU metric gives higher scores to similar sequential words, it can be used as a fluency metric in the designed reward function. The reward function is defined as follow:

$$r_t = \alpha CosineSim(\mathbf{S_0^{'i}}, \mathbf{S'_t}) + \beta BLEU(\mathbf{S_t}, \mathcal{X}) \tag{2}$$

where α controls the contribution of Cosine similarity term, and β controls the contribution of BLEU score.

B. Using Adversaries to Generate Realistic Synthetic News

Up to this point, we have considered the style and topic-preservation. To ensure that the generated news has a similar writing to real news, we use a fake news detection component as an adversary to determine whether the generated news is considered fake or true. Thus, we add an additional term to the reward function:

$$\begin{aligned} r_t =& \alpha CosineSim(\mathbf{S'_0}, \mathbf{S'_t}) + \\ &+ \beta BLEU(\mathbf{S_t}, \mathcal{X}) + \lambda(1 - C_f(\mathbf{S_t})) \end{aligned} \tag{3}$$

Where $C_f \in [0, 1]$ is the confidence of the fake news classifier given an input, and λ shows the importance of this term. The confidence shows the probability of a news content being fake.

For training the agent, we use news dataset $\mathcal{X} = \{(\mathbf{S_0^1}, \mathbf{x_1}), (\mathbf{S_0^2}, \mathbf{x_2})\ \ , ..., (\mathbf{S_0^N}, \mathbf{x_N})\}$ in which $\mathbf{S_0^i}$ shows the topic of i^{th} news and $\mathbf{x_i}$ shows the content of that news. During training, the agent chooses an action a_t leading to selecting word $w_t \in \mathcal{V}$, which is then added to current generated news $\mathbf{S_t} = \{w_1, w_2, ..., w_k, ..., w_t\}$ to generate $\mathbf{S_{t+1}} = \{w_1, ..., w_k, ..., w_t, w_{t+1}\}$. The modified text $\mathbf{S_{t+1}}$ is then passed to the adversary C_f and the reward function to calculate the reward value r_{t+1} considering news content $\mathbf{x}$. Furthermore, the modified text $\mathbf{S_{t+1}}$ is passed to the language model L. Using the outputs of the language model L the environment generates next state s_{t+1}. In the following we discuss the details of using adversarial reinforcement learning.

In adversarial reinforcement learning, the goal is to learn an action policy $\pi(s)$ which leads to maximum amount of accumulated reward $R = \sum_{t=0}^{t=T} r_t$ where T is the terminal time. To find the best action selection policy $\pi(s)$, we use experiences in form of $(s_t, a_t, s_{t+1}, r_{t+1})$ to train the agent. There are different algorithms to train an agent. Policy gradient and Q-Learning are two popular algorithms for training an agent [34]. In this paper we use Deep Q-Learning which is an advanced variant of Q-Learning.

In Deep Q-Learning (DQL), the agent uses a neural network as a function approximator to find an action regarding a given state s. The input of this neural network is state s and the outputs are the values for $(s, a_i)_{i=0}^{|A|}$ where $|A|$ is the number of actions. In DQL, the goal is to learn the following function:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r_{t+1} + \gamma max_{a'} Q^*(s_{t+1}, a')] \tag{4}$$

where Q-function $Q(s, a)$ returns the expected accumulated reward R if the agent selects action a in response to state s and $Q^*(s, a)$ denotes the optimal Q-function which returns the maximum possible accumulated reward R using the optimal policy $\pi(s)$. In this formula, the future rewards are discounted using the γ parameter. We adjust γ with respect to the importance of future rewards.

In practice, it is not feasible to estimate $Q^*(s, a)$ in Equation 4. To overcome this problem, we use a function approximator to estimate the Q-function $Q^*(s, a) \cong Q(s, a; \theta)$. As neural networks are excellent function approximators [35], DQL leverages a neural network with parameters θ called Deep Q-Network (DQN) to find the Q-function $Q(s, a; \theta)$ by minimizing the following loss function:

$$L(\theta) = \mathbb{E}_{s_t, a_t, s_{t+1}, r_{t+1}}[(y - Q(s, a; \theta))^2] \tag{5}$$

where y is the target Q-value calculated using Equation 6:

$$y = \mathbb{E}_{s_{t+1}}[r_{t+1} + \gamma Q(s_{t+1}, a'; \theta')] \tag{6}$$

where θ' is the DQN's parameters from the previous iteration.

Finally, we update the DQN parameters using the derivation of Equation 5 with respect to θ:

$$\begin{aligned}\nabla_\theta L(\theta) =& \mathbb{E}_{s_t, a_t, s_{t+1}, r_{t+1}}[(r + \gamma max_{a'} Q(s_{t+1}, a'; \theta') - \\ &- Q(s_t, a_t; \theta))\nabla_\theta Q(s_t, a_t; \theta)]\end{aligned} \tag{7}$$

In this paper, we have specifically used DQL with memory replay and two networks as target and policy, respectively. The memory replay helps the agent to remember past experiences. The training algorithm is presented in Algorithm 1[1].

IV. Experiments

In this section, we conduct experiments to evaluate the performance of our method. In these experiments, we try to answer the following questions: **Q1:** How well our method can generate synthetic news in comparison to existing methods in terms of topic similarity? **Q2:** How fluent is the generated synthetic news using RLTG? and **Q3:** How well humans evaluate the RLTG's generated synthetic news?

To answer the first question Q1, we consider Cosine similarity; for Q2, we use ROUGE-L metric; and for Q3, we perform human evaluation using a survey to assess RLTG's generated synthetic news in terms of content, title, overall readability, and being realistic or not.

[1]The source code will become publicly available upon acceptance.

Algorithm 1 The Learning Process of RLTG

```
Require: $L$, $\epsilon$, $T$, $M$.
 1: Initialize replay memory $R$, environment, policy, and target networks
 2: while training is not terminal do
 3:     $H, topK \leftarrow L(topic)$
 4:     $s_t \leftarrow M(H, topK)$
 5:     for $t \in \{0, 1, ..., T\}$ do
 6:         Choose action $a_t$ using $\epsilon$-greedy
 7:         Perform $a_t$ on $s_t$ and get $(s_{t+1}, r_{t+1})$
 8:         $R \leftarrow R + (s_t, a_t, r_{t+1}, s_{t+1})$
 9:         $s_t \leftarrow s_{t+1}$
10:         for $(s, a, s', r) \in$ sampled mini-batch $b$ from $R$ do
11:             Update DQN weights using Eq. 7 w.r.t. policy and target networks
12:         end for
13:         if exchange condition met then
14:             Exchange weights between policy and target network
15:         end if
16:     end for
17: end while
```

A. Data

We utilize FakeNewsNet dataset [36] to fine-tune GPT-2 and train our model. This dataset consists of news data $\mathcal{X}$ from two different platforms *GossipCop* and *Politifact*. GossipCop is a fact-checking website, which reports on celebrity news. Politifact is a similar platform, which checks the truth of political news and reports. In this dataset, news are classified into *real* or *fake*. *Politifact* contains $2,645$ true and $2,770$ fake news, while the *GossipCop* includes $3,586$ true and $2,230$ fake news respectively. Dataset statistics are shown in Table I. In this paper, we consider the first few words of each news x_i content as topic $\mathbf{S_0}$.

Platform	G	P
# True news	3,586	2,645
# Fake news	2,230	2,770
# Total News	5,816	5,415

TABLE I: The statistics of the FakeNewsNet dataset

B. Implementation Details

In this part we go through the parameters and implementation details of RLTG. In our model, we use a fine-tuned GPT-2 language model as L. To fine-tune the GPT-2 language model, we first load a pre-trained "GPT-2 medium", then we use FakeNewsNet dataset for 5 iterations to fine-tune the language model. Note that this language model has 12 hidden layers. Each hidden layer returns a tensor with size of (batch size, sequence length, hidden size), where the hidden size in "GPT-2 medium" is 768.

As it is mentioned in the proposed method, the RL agent has a neural network which acts as a function approximator. This network gets a state as input and returns the Q-value for each $(s, a_i)_{i=1}^{K}$ set. This network has 3 layers. The first hidden layer has 1024 nodes, the second and third layer has 512 and 256 nodes, respectively. The output size of this network is equal to the number of actions. In this paper, the number of actions is 50, meaning that the agent chooses between the top 50 words of GPT-2's output probability. The reason we chose 50 is that among values $\{10, 25, 50, 75\}$, it showed a better reward performance than others, with $K = 75$ having a similar performance. Moreover, the the output size of the DQN network is equal to the size of state $\mathbf{s}$. To construct state $\mathbf{s}$, as in Figure 2, we have trained 2 autoencoders and concatenate the output of each encoder to create the state. The first autoencoder is considered for extracting the context of generated news using hidden state H. This autoencoder uses Multi-Layer Perceptron (MLP) to encode and reconstruct the hidden state H. The output of encoder part has 256 nodes. The second autoencoder uses Convolutional Neural Networks (CNN) to extract information about best words positions. The encoder of this autoencoder has an output layer with size of 128. The final size of state $\mathbf{s}$ is 384.

The RL agent is trained on randomly selected topics for 50000 episodes. The agent can choose between top $K = 50$ words from the language model L's output. Each episode has a terminal time of $T = 50$. As the final generated news is more important than the early generated news, a high discount factor $\gamma = 0.9$ is used. As it is mentioned in the proposed method section, we use Deep Q-Learning to train our RL agent. In this algorithm we construct a memory with size 10000 to save the experiences $(s_t, a_t, s_{t+1}, r_{t+1})$. Each experience means that the RL agent chose action a_t in state s_t. The selected action a_t resulted in transition to a new state s_{t+1} and the environment returned a reward r_{t+1}. We then use the memory array to update our model using Equation 5. The batch size for sampling experiences from memory is 32. During the training process we use ϵ-greedy to choose action a_t. This algorithm considers a random action with probability of ϵ and chooses the best action based on Q-values with a probability of $1 - \epsilon$. We use the following decay function to lower the value of ϵ. This function lower the ϵ according to the number of past iterations and exponentially decreases it by a constant rate $\epsilon = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min})e^{\frac{-steps}{decay\_rate}}$ where $steps$ is the number of past iterations and $decay\_rate$ controls how fast the ϵ should decrease. We use $\epsilon_{max} = 0.98$, $\epsilon_{min} = 0.02$ and the decay rate equal to $5,000$.As for the reward function parameters, we set $\alpha = \beta = \lambda = 0.5$. In this case $r \in [0, 1.5]$.

As illustrated in Figure 1, we use a fake news classifier as an adversary to calculate the value of reward function. The architecture of the fake news classifier is shown in Figure 3. The hidden size of bi-directional GRU is 128, resulting in a context vector of 256. The neural network classifier has an input size of 256, hidden size of 128, and output size of 1. We train this classifier before training the agent using Binary Cross Entropy (BCE) loss function. As DQL has a variance during training, we train our model 5 times independently, then we select the agent with the highest average rewards.

To train the RLTG model, we used a publicly available dataset, FakeNewsNet [36] that can be accessed through https://github.com/KaiDMML/FakeNewsNet. To compare RLTG to GPT-2, we have used the Hugging Face package (https://huggingface.co) to use and fine-tuned the GPT-2 model. The GPT-2 model is fine-tuned for 2 iterations using the FakeNewsNet dataset. Moreover, to run the experiments on Grover [2], we used their publicly available package through https://rowanzellers.com/grover.

C. Experimental Design

We use different baselines for comparison. As our proposed model is based on the OpenAI's GPT-2 language model, we use this language model alone as a baseline to determine how using an RL agent on this model can improve its results. Furthermore, we also include the RL agent alone as a baseline to generate synthetic news. Our main goal, generating synthetic news content, is close to the Grover [2], so we have selected this work as a baseline. Finally we select the SeqGAN method because it incorporates GAN with reinforcement learning. Following is the description of the baselines:

- **GPT-2 [5]:** a language model capable of generating long text. This language model is based on transformers and has three different variations based on it's number of layers and parameters: small (117M parameters) , medium (345M parameters), and large (774M parameters). In this paper we use GPT-2 medium as fine-tuning it needs less resources.
- **Fine-tuned GPT-2 (FTGPT-2):** similar to GPT-2, but has been fine-tuned using FakeNewsNet dataset.
- **RL:** in this baseline we use RL technique without using a language model to train an agent. In this case, all components except the *actions* are the same as the proposed RLTG method. The action set in this baseline is all word in the vocabulary set V. The training process of this baseline is similar to our model.
- **Grover [2]:** a conditional language model which can generate text based on given parameters: domain, date, authors, and headline. The goal of Grover is to generate news content based on different parameters. While the results are promising, it seems this language model is very dependent on *domain* parameter, which we will explore during our evaluation.
- **SeqGAN (SeqG) [15]:** is a text generation method, which models data generator as a stochastic policy in reinforcement learning. They then use policy gradient method to train their model.
- **PPLM [12]:** is a method that leverages GPT-2 to generate domain-specific text. We have used our pre-trained fakenews classifier to generate fakenews related text.

D. Experimental Results

We evaluate our model's performance regarding Q1 - Q3.

Topic Similarity (Q1). To answer this question, we use cosine similarity as in Equation 1 to calculate the similarity between the embedding of the given topic $\mathbf{S_0}$ and the generated news $\mathbf{S_T}$. We do not want the RL agent to exactly select topic words to maximize its reward, so we use text embeddings to calculate the similarity. In this case, the agent chooses words to maximize the context similarity between both the topic and the generated text. For a fair comparison, we use a fixed sentence length of 200 for text generation.

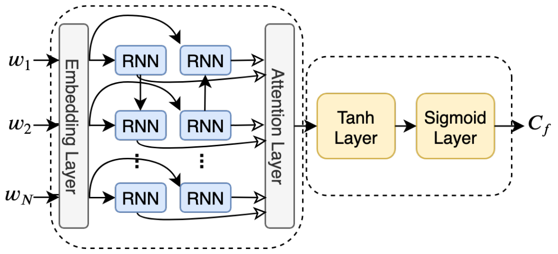


Fig. 3: The architecture of fake news classifier. It uses bi-directional GRU and an attention layer to create context vector, then uses a three layer neural network to classify the news as fake or true.

Table II shows the performance of RLTG against other baselines. RLTG can outperform other baselines because it considers topic similarity during the training process. Although fine-tuned GPT-2 falls behind RLTG and Grover, it has achieved a high similarity comparing to other methods. This is due to the fact that the fine-tuned GPT-2 tends to repeat itself. Note that the performance of the RL baseline is behind all models. The reason behind it is that the action set in this case is very large and the agent cannot converge easily. Furthermore, using a language model to narrow down the possible actions can have a huge impact on training the model. Finally, comparing the performance of RLTG and baselines to PPLM shows that PPLM lacks fluency. We suspect this is due to the used classifier and the fact that the GPT-2 used in this model cannot generate text in the news domain.

Fluency Test (Q2). To answer this question we use both perplexity and ROUGE-L metrics. Perplexity may not be suitable for showing the effectiveness of a model in open-domain text generation [37], but in our case, we focus on domain-specific news generation. Table II shows the results for fluency test. Lower perplexity means the generated news is more concentrated and it is less variant. Furthermore, the ROUGE-L score applies Longest Common Subsequence between the news contents $\mathcal{X}$ and generated news content $\mathbf{S_T}$ to calculate the final score. We select ROUGE-L metric for evaluation because our method is trained to achieve a high BLEU score, and using BLEU score for evaluation is not fair. ROUGE-L measures how many words from the reference sentences have appeared in the generated news. ROUGE-L gives higher scores to sequential words, and can be used as a fluency metric. In this paper we use the FakeNewsNet dataset as the reference sentences. A higher ROUGE-L score means the generated news is more fluent.

	RLTG	GPT-2	FTGPT-2	Grover	SeqG	PPLM	RL
Similarity	**0.342**	0.176	0.241	0.313	0.301	0.254	0.153
Perplexity	**14.8**	22.3	19.8	15.3	17.4	21.8	30.4
ROUGE-L	**28.4%**	23.1%	24.6%	27.3%	21.5%	18.6%	17.2%

TABLE II: Topic similarity (↑ better), perplexity (↓ better), and ROUGE-L score (↑ better) for model's generated news.

Human Evaluation (Q3). To further investigate the quality of the generated text, we conduct a human study to evaluate the generated news without knowing the origin of the news (human or machine generated). In this human study, we asked the participants to give a score from 1 to 3 about topic similarity, writing style, content quality, and overall evaluation of the given text. Table III shows the designed questionnaire for evaluating the performance of the language models. We used a similar measure as [2] and included a new question regarding the topic similarity.

#	Measure	Question
1	Realistic	Is the style of this article consistent? **(3)**. Yes, this sounds like an article I would find at an online news source. **(2)**. Sort of, but there are certain sentences that are awkward or strange. **(1)**. No, it reads like it's written by a madman.
2	Content	Does the content of this article make sense? **(3)**. Yes, this article reads coherently. **(2)**. Sort of, but I don't understand what the author means in certain places. **(1)**. No, I have no (or almost no) idea what the author is trying to say.
3	Title	Does the article sound like it's around a topic? **(3)**. Yes, I feel that this article is talking about a single topic. **(2)**. Sort of, I'm not sure what the article is about. **(1)**. No, it seems this article is gibberish.
4	Overall	Does the article read like it comes from a trustworthy source? **(3)**. Yes, I feel that this article could come from a news source I would trust. **(2)**. Sort of, but something seems a bit fishy. **(1)**. No, this seems like it comes from an unreliable source.

TABLE III: Human evaluation questionnaire

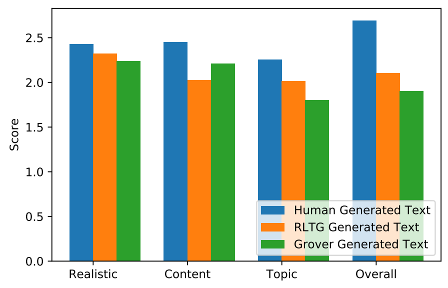
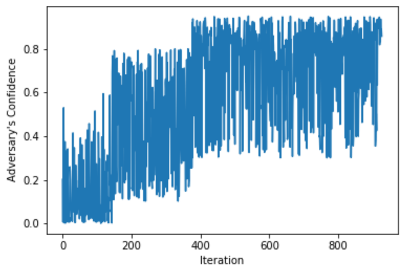


Fig. 4: Human evaluation results. Each participants evaluated each articles based on its style, topic similarity, content quality, and overall evaluation. Higher score is better.

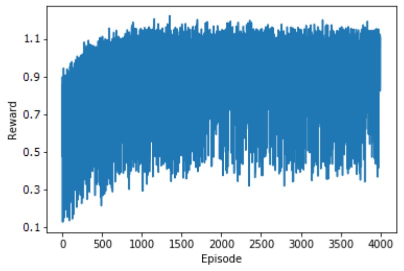
For comparison, we consider best performing models, RLTG and Grover. We include 75 articles (25 human generated, 25 RLTG generated, and 25 Grover generated). The results are provided in Figure 4.

E. Parameter Analysis

In this part, we study the effects of different parameters of our model on the quality of the generated news. These parameters include the reward function's parameters, α, β, and γ. The α parameter indicates the importance of topic similarity, β shows the importance of readability according to the BLEU score, and γ shows the importance of the adversary which is the fake news classifier.

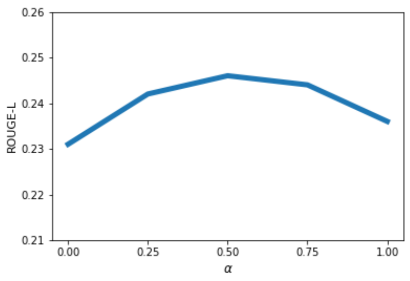


(a) **The reverse confidence of fake news classifier $(1 - C_f)$ from different episodes during training.**

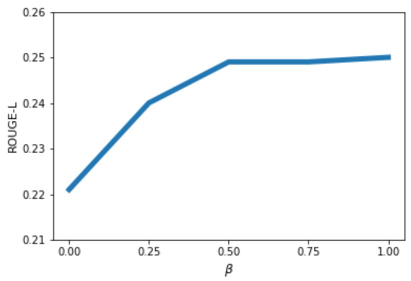


(b) **Average of agent's reward from episodes 1 to 4000.**

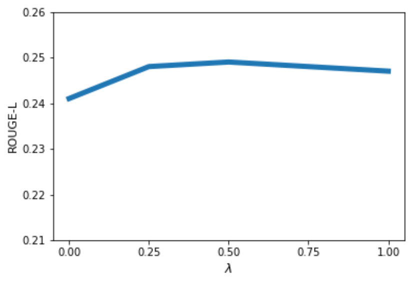
Fig. 5: The reverse of adversary's confidence $(1 - C_f)$ and RL agent's rewards show the learning process.



(a) **Impact of α**



(b) **Impact of β**



(c) **Impact of λ**

Fig. 6: Impact of different reward function parameters on RLTG's ROUGE-L score.

Furthermore, in Figure 5 we show the reverse confidence of classifier $(1-C_f)$ of the fake news classifier for several periods of training iterations. This figure shows the confidence for the final generated news at terminal time T. From this figure we conclude that the agent can generate realistic fake news.

First, we assess the convergence of the training algorithm by showing the RL algorithm's reward values during the training. Figure 5 shows the mean reward for each episode over for each iteration. The results indicates that the average reward of agent is increasing over time, meaning that the agent is learning a policy $\pi(s)$ which can result in larger reward values. At first the rewards are low which is as a result of randomness during early episodes, but it increases as the agent learns better actions for each state s. Note that in Figure 5 we only show the reward values for first 4000 iterations to show its increasing behavior.

To measure the impact of different components in RLTG, we perform two different tests. On a high level, RLTG is composed of two main components, the GPT-2 language model, and the RL agent. In the first test, we study the performance of each high level component, RL and GPT-2, in RLTG. Table II indicates the performance for RL, GPT-2, and RLTG. We notice that the RL model performs poorly which is expected due to the GPT-2's superior architecture. In the second test, we study the importance of different components in the RL part of the RLTG by analyzing the effects of α, β, and γ on the reward function in Equation 3. The effect of these parameters is illustrated by changing them as $\alpha, \beta, \lambda \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ and calculating the ROUGE-L score.

Figure 6 shows the effect of each component on the ROUGE-L score. For each parameter, we consider other parameter values as 0.5. While α and λ have little effect on the ROUGE-L score, the β parameter has a larger impact as it considers how well the generated news content overlaps with the given domain according to dataset $\mathcal{X}$.

We further study the effect of using a fake news detection classifier as an adversary to see if the generated news is realistic enough not to be detected as fake. By studying the reward values in Figure 5, we can see that the agent can generate news content which the adversary cannot easily detect as fake. The trained adversary has an accuracy of $\%81.3$ and AUC of $\%75.3$. Although it seems that the γ parameter does not contribute to generating a higher-quality text, according to the confidence values presented in Figure 5, RLTG can avoid producing obvious fake synthetic news by using features that bypasses the adversary.

F. Case Studies of Generated Synthetic News

For illustration purposes, we show some examples of the synthetic news generated by RLTG and by the baselines. Table IV shows two examples of generated news by different models. The topic-related words are highlighted to show how well RLTG preserves the given topic in the generated synthetic news. Finally, Table V illustrates the results of the human evaluation experiment for five generated synthetic news.

Model	Text
RLTG	**the wedding of prince harry and meghan markle** was held with *royal* *family* *guests* attending the *ceremony*. After nearly four hours on site, they met again with *wedding* *guests* from a host community to gather for one particular special day. In May there would been even-clothing on all royal guests arriving as many couples wore the customised *royal* gown...
FTGPT-2	**the wedding of prince harry and meghan markle** was held on the grounds of St. George's Chapel in Windsor, England, on May 19, 1814. The bride and *groom* were Prince William and Kate Middleton, and the *groom* was Prince Charles. The *bride*...
GPT-2	**the wedding of prince harry and meghan markle** was held on the *wedding* of prince harry and meghan markle was held on the day of the *wedding* of prince harry and meghan markle was held on the day...
Grover	**the wedding of prince harry and meghan markle**. Like many of the *gala* trends we *celebrate* today, the *wedding* of prince harry and meghan markle has taken center stage at the MWC. The mom and pop couples still roping in floral designs, staging weddings, and of course, speaking on stage at the hands of this mortal pair.
RLTG	**kim kardashian and kanye west are reportedly getting married in florence**, *Italy*! Kim *confirmed* it this evening after receiving numerous texts during Sunday. We will certainly keep up this morning's *news*, as long as this story reaches our lips...
FTGPT-2	**kim kardashian and kanye west are reportedly getting married in florence**. The *couple*, who have been dating since 2015, have been spotted *together* in the city of L.A. and are reportedly spending time *together*. The *couple*, who have been dating since 2015, have been spotted *together* in the city of L.A. and are reportedly spending time *together*...
GPT-2	**kim kardashian and kanye west are reportedly getting married in florence**. The *couple* is reportedly planning to have a baby *together*. The couple is reportedly planning to have a baby...
Grover	**kim kardashian and kanye west are reportedly getting married in florence**. According to BeBe, whose team is developing ready-made phones. Kim Kardashian West and Kanye West have *called* it quits on their *engagement*. "Kim Kardashian and Kanye West announce divorce...

TABLE IV: Sample generated news given a news topic. The topic is in **bold** text and the *highlighted* words are the most related words to the topic according to their representations' cosine similarity.

Generated news (The **bold** part is the given topic)	Mean score
The star who was accused of rape late last year is getting on an international show: "because I will bring awareness by supporting our families from India for this year on and up! I need every bit it might. this week has turned one world back". Pranta had previously made some serious noises over India-friendly topics ...	2.33
meghan markle and prince harry on monday: the british royal family are not welcome. "They do not like me, but are happy that we got married. we can be proud and have some nice holidays together."	2.33
share fans of netflix's cult favorite scifi drama series sense8 have a very special gift of nostalgia from them that can never truly forget. a beloved franchise in some small but undeniable shape. So this, we know how fans would look after that they would not enjoy being seen. it will become apparent soon when people who enjoy being on such shows, may also like to share these experiences on facebook to have them see those great and unique stories.	2.25
jade is my first friend to ever be pregnant in her mid 30ies. I can feel good about the baby! she will make an excellent wife. I just don't expect a kind person or family who wants the support of an anesthesia for that to occur to a man with diabetes.	2.00
imdbcom inc takes no responsibility for the content or accuracy of its claims. It has taken place today (August 21), when an article appeared at Wired, which says: "there will be one man with guns on their faces who at a certain level who can take on ISIS". It has gone into further information on what constitutes terrorist threats (or just about all terrorist activity).	2.50
robert pattinson says he was just kidding around about being asked by anorexist about a possible relationship, "so we were like, I don't know if he's a guy that I want. We're just trying something out, and we just don't have that. So he said I don't want that. He was like a little boy and I don't want it." I have no plans that are to go into this story.	2.67

TABLE V: Sample generated news and their average human evaluation scores given a news topic using RLTG.

V. Conclusion and Future Work

Text generation is crucial and can be used in different NLP applications. One application of text generation is news content generation. In this paper we challenge the problem of generating realistic topic-preserving news by leveraging a pre-trained language model. To this end, we propose a reinforced model RLTG to control a language model toward news content generation. This model uses Deep Q-Learning to train an agent capable of selecting words from a language model's output to generate realistic topic-preserving news. There are various future directions that can benefit from this work. One future direction is to study different types of biases in the generated news and create a de-biasing model that removes bias and unfairness from pre-trained text generation models. Most importantly, we can study and analyze the hidden features and differences between real and synthetic news content. Because language models are advancing quickly and they are capable of generating believable fake news, this work can help us to detect machine generated fake news in future.

VI. Ethics Statement

This work aims to advance research efforts in synthetic news generation, a topic that has yet to be studied extensively. Here, considering current solutions, much work remains to elucidate how to build an effective news generation model. We believe that generating synthetic news is a stepping stone that enable us to further investigate and detect machine-generated fake news. We are committed to preventing fake news on social media. To this end, we plan to release a limited version of our trained model that can be only used for specific topics. Moreover, we will share a large set of generated news using our model that researchers can use in their experiments.

VII. Acknowledgements

This material is based upon the work supported, in part, by ONR N00014-21-1-4002 and AFRL FA8650-15-D-6583/FA8650-17-F-6820. Kai Shu is supported by the NSF award #2109316.

References

[1] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[2] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," *arXiv preprint arXiv:1905.12616*, 2019.

[3] H. Allcott and M. Gentzkow, "Social media and fake news in the 2016 election," *Journal of economic perspectives*, vol. 31, no. 2, pp. 211–36, 2017.

[4] 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.

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019.

[6] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[7] A. Fan, M. Lewis, and Y. Dauphin, "Strategies for structuring story generation," *arXiv preprint arXiv:1902.01109*, 2019.

[8] R. Puduppully, L. Dong, and M. Lapata, "Data-to-text generation with content selection and planning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6908–6915.

[9] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," in *Advances in Neural Information Processing Systems*, 2017, pp. 3155–3165.

[10] W. Fedus, I. Goodfellow, and A. M. Dai, "Maskgan: better text generation via filling in the_," *arXiv preprint arXiv:1801.07736*, 2018.

[11] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1587–1596.

[12] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, and R. Liu, "Plug and play language models: a simple approach to controlled text generation," *arXiv preprint arXiv:1912.02164*, 2019.

[13] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[14] F. Huszár, "How (not) to train your generative model: Scheduled sampling, likelihood, adversary?" *arXiv preprint arXiv:1511.05101*, 2015.

[15] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, "Adversarial feature matching for text generation," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017.

[22] K. Wang and X. Wan, "Sentigan: Generating sentimental texts via mixture adversarial networks." in *IJCAI*, 2018, pp. 4446–4452.

[23] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep reinforcement learning for dialogue generation," *arXiv preprint arXiv:1606.01541*, 2016.

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

[25] Z. Shi, X. Chen, X. Qiu, and X. Huang, "Toward diverse text generation with inverse reinforcement learning," *arXiv preprint arXiv:1804.11258*, 2018.

[26] Z. Fan, Z. Wei, S. Wang, Y. Liu, and X.-J. Huang, "A reinforcement learning framework for natural question generation using bi-discriminators," in *Proceedings of the 27th ICCL*, 2018, pp. 1763–1774.

[27] P. Ke, F. Huang, M. Huang, and X. Zhu, "Araml: A stable adversarial training framework for text generation," *arXiv preprint arXiv:1908.07195*, 2019.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[29] S. C. AP, S. Lauly, H. Larochelle, M. Khapra, B. Ravindran, V. C. Raykar, and A. Saha, "An autoencoder approach to learning bilingual word representations," in *Advances in neural information processing systems*, 2014, pp. 1853–1861.

[30] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," *arXiv preprint arXiv:1606.01781*, 2016.

[31] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.

[32] P. Budzianowski and I. Vulić, "Hello, it's gpt-2–how can i help you? towards the use of pretrained language models for task-oriented dialogue systems," *arXiv preprint arXiv:1907.05774*, 2019.

[33] 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*. ACL, 2002, pp. 311–318.

[34] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[35] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[36] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, "Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media," *arXiv preprint arXiv:1809.01286*, 2018.

[37] C.-W. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau, "How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation," *arXiv preprint arXiv:1603.08023*, 2016.