University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization:	Spring semester, 20...... Open / Restricted access
Writer:	…………………………………………. (Writer's signature)
Faculty supervisor: External supervisor(s):	
Thesis title:	
Credits (ECTS):	
Key words:	Pages: ………………… + enclosure: ………… Stavanger, ……………….. Date/year

University of Stavanger

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Generative adversarial networks for bias flipping

Master's Thesis in Computer Science
by

Khoa Nguyen Le

Internal Supervisors

Vinay Jayarama Setty, Department of Computer Science and Electrical Engineering, University of Stavanger

June 30, 2020

"Our intelligence is what makes us human, and AI is an extension of that quality."

Yann LeCun

Abstract

The disinformation news in media channels such as social media websites or online newspapers has become a big challenge for many organizations, governments, and scientific researchers. In connection to fake news, the political bias (left-wing or right-wing) of the news articles are recently receiving more attention. In this thesis, we leverage the Adversarially Regularized AutoEncoder (ARAE) model, which enhances the adversarial autoencoder (AAE) by learning a parameterized prior as a Generative Adversarial Networks (GAN) to generate bias-flipped headlines. We perform the experiments with multiple datasets then discuss how these approaches contribute to the bias flipping and detecting problems.

Acknowledgements

First, I would like to thank advisor Vinay Jayarama Setty for giving me lots of advice, datasets, and also computing resources throughout the thesis project. I would also like to thank Nina Egeland for her administrative support, especially during the COVID-19 pandemic.

Second, I thank the technical team at the University of Stavanger for allocating the GPU servers to train our models.

Khoa Nguyen Le, Stavanger, June 2020.

Contents

Abstract **vi**

Acknowledgements **viii**

Abbreviations **xi**

Symbols **xiii**

1 Introduction **1**

2 Related Work **3**

2.1 Text generation using GAN 3

2.2 Text style transfer 3

2.3 Bias flipping 4

3 Background **5**

3.1 Artificial Neural Networks (ANN) 5

3.1.1 Activation function 6

3.1.2 Dropout 6

3.1.3 Adaptive Moment Estimation (Adam) 6

3.2 Recurrent Neural Networks (RNN) 7

3.3 Long Short Term Memory (LSTM) 8

3.4 Convolutional Neural Network (CNN) 9

3.5 Autoencoder 9

3.6 Generative Adversarial Networks (GAN) 9

3.7 Embeddings 11

3.7.1 Word2vec 11

3.7.2 Global Vector 11

3.8 Text Generation Evaluation Metrics 12

3.8.1 Bi-Lingual Evaluation Understudy (BLEU) 12

3.8.2 Recall-Oriented Understudy for Gisting Evaluation (ROUGE) . . . 12

4 Data **13**

4.1 The Webis Bias Flipper 2018 Dataset 13

4.2 Hyperpartisan News Dataset 14

5 Solution Approach **17**
5.1 Existing Approaches/Baselines 17
5.2 Analysis 19
5.3 Proposed Solution 20
5.3.1 Adversarially Regularized Autoencoder 20
5.3.2 ARAE for bias flipping 22
5.3.3 Vocabulary 22
5.3.4 Implementation 23

6 Experimental Evaluation **25**
6.1 Experimental Setup 25
6.1.1 Model setup and parameters 25
6.1.2 Evaluation metrics 27
6.2 Experimental Results and Discussions 27
6.2.1 Webis Bias Flipper 2018 27
6.2.2 Hyperpartisan News 29

7 Conclusion and Future Directions **33**
7.1 Conclusion 33
7.2 Future Directions 33
7.2.1 Perform more experiments on hyperparameters tuning 33
7.2.2 Train with other datasets 34
7.2.3 Leverage the critic of the ARAE model to use as a Political Bias Classifier 34
7.2.4 Automatic evaluation metrics 34

List of Figures **34**

List of Tables **37**

Bibliography **39**

Abbreviations

Acronym	**W**hat (it) **S**tands **F**or
Adam	**Ada**ptive **m**oment estimation
AE	**A**uto**E**ncoder
ANN	**A**rtificial **N**eural **N**etworks
ARAE	**A**dversarially **R**egularized **A**uto**E**ncoder
BLEU	**B**i-**L**ingual **E**valuation **U**nderstudy
CNN	**C**onvolutional **N**eural **N**etwork
CPU	**C**entral **P**rocessing **U**nit
D	**D**iscriminator
DNN	**D**eep **N**eural **N**etworks
G	**G**enerator
GAN	**G**enerative **A**dversarial **N**etworks
GloVe	**Glo**bal **Ve**ctor
GPU	**G**raphics **P**rocessing **U**nit
MLP	**M**ulti-**L**ayer **P**erceptron
NLP	**N**atural **L**anguage **P**rocessing
NLTK	**N**atural **L**anguage **T**ool**k**it
LSTM	**L**ong **S**hort **T**erm **M**emory
OS	**O**perating **S**ystem
ReLU	**Re**ctified **L**inear **U**nit
RNN	**R**ecurrent **N**eural **N**etworks
ROUGE	**R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation
SGD	**S**tochastic **G**radient **D**escent
VAE	**V**ariational **A**uto**E**ncoder
WAE	**W**asserstein **A**uto**E**ncoder

Symbols

symbol	name
b	bias
c	cell
f, g	activation function
h	hidden state, hidden layer
p	probability
s	sentence, noise
x	input data
y	output
z	encoding, input content
$\mathbb{E}$	cross entropy
$\mathcal{L}$	loss function
$\mathbb{P}_Q$	distribution of encoding
$\mathbb{P}_{\mathbf{z}}$	prior distribution
$\mathbb{P}_*$	real distribution
W	weight matrix
σ	sigmoid function
ϕ	encoder parameters
ψ	decoder parameters
θ	generator parameters

Chapter 1

Introduction

To date, media bias and fake news can be found on any media outlets, especially social media like Facebook, Twitter, and online newspapers. Thus, there's an increasing number of public organizations and scientific researchers trying to tackle this problem [1].

In this thesis, our approach puts more focus on the writing style of the news articles, in particular, political-biased headlines and claims. The work from [2] have shown that the articles' writing styles could be used to detect the hyperpartisan news from the more balanced ones. The style of left-oriented and right-oriented articles appear to be more related than they have with mainstream's style or to the opposite side's style. Considering an example from [3]:

Why Trump is right in recognizing Jerusalem as Israel's capital

Trump is making a huge mistake on Jerusalem

The two headlines above are from Fox News and New York Times respectively and are about the same event of Donald Trump recognizing Jerusalem as the Israel's capital. The different stances in those headlines can lead the readers to very different impressions.

In order to learn the style of news headlines or claims, we perform unaligned textual style transfer using the adversarially regularized autoencoder (ARAE) [4]. This model is based on the Wasserstein autoencoder (WAE) framework [5] and generative adversarial networks (GAN) [6] which has contributed a major advancement in text generation tasks. We also use GloVe [7] pre-trained vectors as word embedding to replace for the ARAE model's word embedding which only relied on the word indices in the vocabulary.

The experiments have been performed on two datasets:

- The Webis Bias Flipper 2018 [8] comprises 2781 events from allsides.com as of June 1st, 2012 till February 10, 2018. We use this dataset to generate the right-oriented and left-oriented bias articles titles.
- The Hyperpartisan News dataset [9] contains 750,000 articles labeled by the overall bias of the publisher and 645 articles labeled through crowdsourcing on an article basis.

The contribution of this thesis is applying an advanced model of generative adversarial networks to transfer the news title into opposite bias and examine if the model can improve the quality of generated headlines as well as neutralize the hyperpartisan news articles.

In the remainder of the thesis, some related work and background are covered in the next two chapters. Chapter 4 will present the datasets and chapter 5 goes to details of the solution approach. Then the experimental results will be reported and discussed in chapter 6. Finally, chapter 7 will conclude the paper as well as suggest some future research directions.

Chapter 2

Related Work

This chapter briefly highlights the related work done in the domain of text generation, bias flipping on news articles and text style transfer.

2.1 Text generation using GAN

Text generation is a common task in many NLP applications. The paper *Texygen: A Benchmarking Platform for Text Generation Models* [10] has shown benchmarking results of multiple GAN methods for text generation in which LeakGAN [11] obtains the highest score. LeakGAN allows the discriminator leak information from the generated text to help the generative network achieve better performance. Those GAN models often need pre-training which is computationally expensive. A recent approach has proposed a new solution called TextKD-GAN [12] using GAN to generate text with knowledge distillation [13]. This model is also based on autoencoder (AE) to create text representation. TextKD-GAN has a comparable performance without the need for pre-training.

2.2 Text style transfer

In the line of text generation, text style transfer is a task that generates the new sentences with style is different from the source sentences while keeping the same semantic. This task is hard due to the lack of a parallel training dataset [14]. One approach of those text style transfer methods is dividing the style and content from the sentences. The variational autoencoder (VAE) has been used in [15] to change the style of a given sentence to the target style aspect. [16] presents the aligned AE and cross aligned AE to modify the sentiment in Yelp reviews.

2.3 Bias flipping

The bias flipping task is early proposed in the paper *Learning to Flip the Bias of News Headlines* [3]. They build a corpus to analyze the bias in political news and use the cross-aligned AE model from [15] to generate the headlines with flipped bias. Our experiments are also performed on this provided dataset. Although they indicate that the bias may only appear on different levels from a single sentence to paragraph and the whole article, they only apply the bias flipper for sentence-level or news headline in particular. In this thesis, we also focus only on the sentence level to do the bias flipping.

Chapter 3

Background

This chapter introduces the theory and background knowledge in brief that is needed to explain the thesis's approach, as well as the evaluation metrics.

3.1 Artificial Neural Networks (ANN)

An Artificial Neural Networks (ANN), also known as *multi-layer perceptron* (MLP), is a collection of *artificial neurons* [17] which models the neurons in a human brain. A artificial neuron can be presented as the following mathematical function:

$$y = f(Wx + b) \tag{3.1}$$

where x is the input, W is the weight matrix, b is the bias and y is the output. The *activation function* f is typically a non-linear function. The range of f is finite and often is (0, 1) or (-1, 1). A *training* process of an ANN is the process to adjust W and b values in order to output y that's reasonable to a set of input x. A common approach to train an ANN and update the weights and biases is Backpropagation algorithm [18]. ANN can be formed by one *input layer*, one *output layer* and multiple *hidden layers* of artificial neurons. Figure 3.1 illustrates a simple artificial neural networks.

The ANNs with a huge number of hidden layers can form a more powerful model, also referred to Deep neural networks (DNN). More advanced networks can also contain multiple input layers and/or output layers.

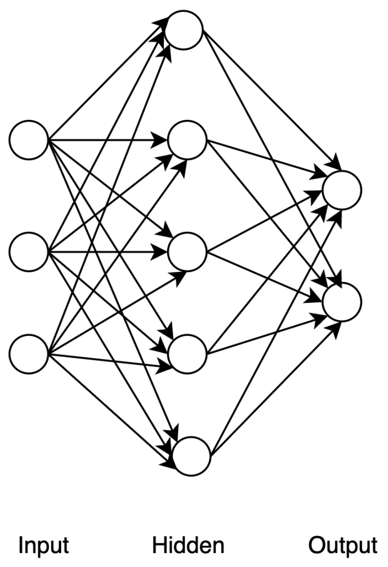

Figure 3.1: A simple ANN

3.1.1 Activation function

Generally, the *activation functions* are non-linear. *Sigmoid* [19] functions are the famous choices for activation function. Some examples:

Logistic function: $f(x) = \frac{1}{1+e^{-x}}$

Hyperbolic tangent function: $f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Recently, ReLU [20] family functions (ReLU, PReLU, Leaky ReLU, RReLU [21]) have become most commonly used as the activation function in neural networks since it has an efficient computational cost and it's better when combined with gradient propagation with less vanishing gradient problems.

3.1.2 Dropout

When an ANN performs very well on training data but gets bad results with test data, that's a sign of *overfitting*. *Dropout* [22] is a simple but effective way to prevent neural networks from overfitting. It generalizes the networks by activating neurons based on a probability.

3.1.3 Adaptive Moment Estimation (Adam)

Adaptive Moment Estimation (Adam) [23] is an optimization algorithm that can be used to update the weights parameter during the training process and obtain good results fast. Adam is an extension to *stochastic gradient descent (SGD)* [24]. It leverages the momentum by using the moving average of the gradient while SGD with momentum only uses the gradient itself. Adam is computationally efficient and well appropriate for models with a huge number of parameters or large datasets.

3.2 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNN) networks are a type of artificial neural network specifically used to find patterns in a sequence of data. The network has recurrent connections between layers that allow it to handle a data sequence. Figure 3.2 illustrates a simple RNN where $x_t, t = 1, 2, ..., T$ is the input sequence, y_t are the output and h_t are the hidden states at each time step t.

The value of hidden state h_t is computed as a linear combination, denoted by $\oplus$, of the previous hidden state h_{t-1} and the input token x_t:

$$h_t = g(W(x_t \oplus h_{t-1})) \tag{3.2}$$

where g is an activation function, such as *sigmoid* or *tanh*, and W is the weight matrix.

RNN is useful dealing with text generation problems. In order to generate a sequence $y_1, y_2, ..., y_t$, RNN computes the conditional probability for each token, given the previous selected tokens and select the token with highest probability:

$$p(y_t|y_{t-1}, ..., y_1) = g(h_t) \tag{3.3}$$

One disadvantage is that as the sequence grows larger, RNN gets difficulties to predict the next element in the sequence. Therefore, the generated text lacks real meaning.

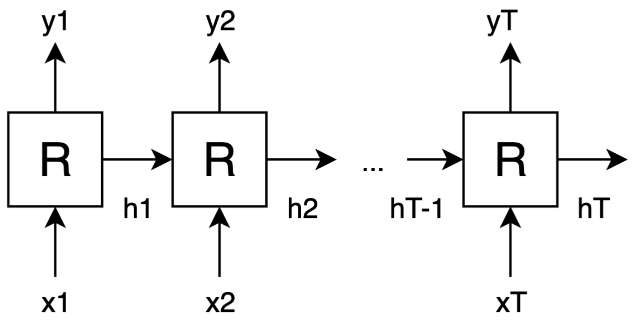

Figure 3.2: Illustration of a RNN

3.3 Long Short Term Memory (LSTM)

Long Short Term Memory model (LSTM) [25], introduced by Hochreiter and Schmidhuber, is a special type of RNN. Unlike original RNN, it can capture the potential long-distance dependencies. LSTMs have four neural network layers and they are, basically, uni-directional and receive information from past states only.

For more details, RNN's units are replaced with more sophisticated units in LSTM networks. LSTM unit has three independent gates: *input gate* i_t, *output gate* o_t and *forget gate* f_t:

$$i_t = \sigma(W_i[x_t; h_{t-1}] + b_i) \tag{3.4}$$

$$o_t = \sigma(W_o[x_t; h_{t-1}] + b_o) \tag{3.5}$$

$$f_t = \sigma(W_f[x_t; h_{t-1}] + b_f) \tag{3.6}$$

LSTM unit also has a *memory cell* c_t that can be considered as the internal memory of the unit. It's state value depends on the previous memory cell state c_{t-1} and a candidate new value of the memory cell $\tilde{c}_t$:

$$\tilde{c}_t = tanh(W_c[x_t; h_{t-1}] + b_c) \tag{3.7}$$

$$c_t = f_t c_{t-1} + i_t \tilde{c}_t \tag{3.8}$$

The hidden state at the time step t is computed as follows:

$$h_t = o_t tanh(c_t) \tag{3.9}$$

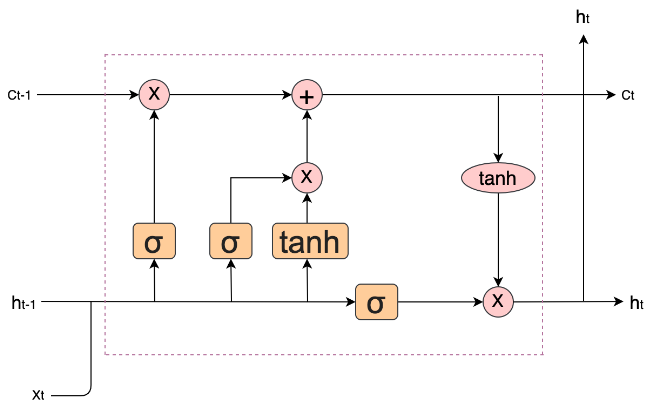

Figure 3.3: Illustration of a LSTM unit

Figure 3.3 shows the architecture of a LSTM unit.

3.4 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a type of ANN that includes convolutional layers in the hidden layers [26]. The filters in the convolutional layers can reduce the dimension of data between layers and produce the output. CNN has become a popular choice when building a discriminative model. Yoon Kim [27] proposed the usage of CNN in the context of sentence classification, which was previously developed for computer vision. Even though CNNs may lose some vital context information, they still are a powerful tool for feature representation. In this thesis, we also leverage a simple bias classifier that is built on top of CNN as a part of the evaluation for the generated text. This will be described in chapter 6.

3.5 Autoencoder

Autoencoder is an ANN type used to learn a representation of data in *unsupervised learning* [28]. An autoencoder consists of an *encoder* that computes the input into a reduced encoding, and a *decoder* that reconstruct the encoding into representation as close as possible to the input. Figure 3.4 illustrate a simple autoencoder architecture. A variant of Autoencoder, *Variational autoencoder*, is recently become popular to solve the generative problems [29].

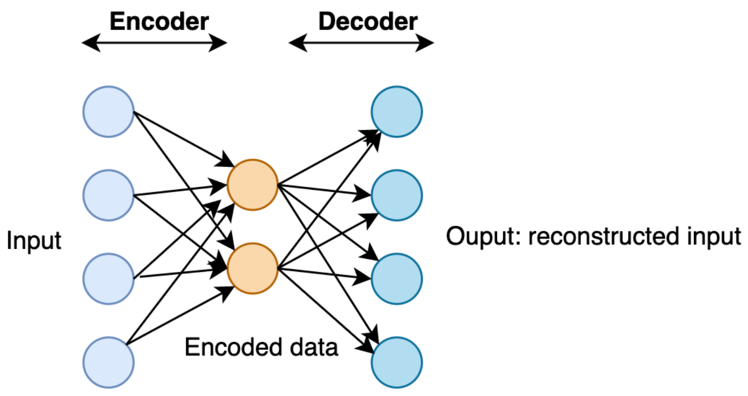

Figure 3.4: Simple Autoencoder architecture

3.6 Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GAN) was introduced by Goodfellow et al in 2014 [6]. A GAN model consists of one *generator network G* that is trained to generate realistic samples and one *discriminator network* D to detect if a given sample is from the training data or generated by G. The model can be simply described by Figure 3.5. GAN can be considered as a *minimax game*:

$$\mathcal{L}_{GAN} = \mathcal{L}_G = -\mathcal{L}_D \tag{3.10}$$

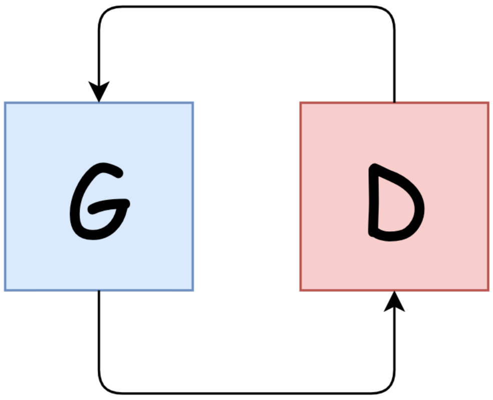

Figure 3.5: GAN general architecture illustration, inspired by [30]

where L_G and L_D are the cost function of the generator and discriminator respectively. The loss function can be defined as follow using *cross entropy* [31]:

$$\mathcal{L}_{GAN} = \mathbb{E}_{x \sim p_r} logD(x) + \mathbb{E}_{z \sim p_g} log[1 - D(G(z))] \tag{3.11}$$

where p_r is the distribution of real data and p_g is the distribution of generated data (the notations are taken from [30]). The discriminator parameters θ^D are then updated using the gradient

$$\nabla_{\theta^D} \mathcal{L}_D(\theta^D, \theta^G) = -\nabla_{\theta^D} \frac{1}{N} \sum_{n=1}^{N} [logD(x^n) + log(1 - D(\tilde{x}^n))] \tag{3.12}$$

where θ^G denotes for generator parameters, samples x are from p_r and $\tilde{x} = G(z)$ are from the generated data distribution $p_g(z; \theta^G)$. The generator G will be updated after by stochastic gradient descent:

$$\nabla_{\theta^G} \mathcal{L}_G(\theta^G) = \nabla_{\theta^G} \frac{1}{N} \sum_{n=1}^{N} log(1 - D(\tilde{x}^n)) \tag{3.13}$$

Theoretically, both generator and discriminator are continuously improved until p_g converges to p_r. One disadvantage of GAN is that there's no accurate representation for the distribution of generated data p_g. Another disadvantage is that if D training and G training processes are not balanced (G is trained too much without updating D), then many generated values of z will be collapsed to the same value of x and G could not be able to model the real data distribution [6].

GANs have shown high effectiveness for image generation and recently have been applied in text generation with promising results [32] [11].

3.7 Embeddings

Before we perform NLP tasks such as text classification or generation, words and documents should be represented as numeric vectors.

3.7.1 Word2vec

Word2Vec gives a numeric representation for each word in the text corpus. It's able to capture the different relations between words, such as synonyms, antonyms, or analogies (for example: vector("King") - vector("Man") + vector("Woman") = vector("Queen")) [33]. There're two popular Word2vec models: Continuous Bag-of-Words (CBOW) and the Skip-Gram. The CBOW model predicts the current word based on the surrounding words (context), and the Skip-gram model predicts context given the current word [33].

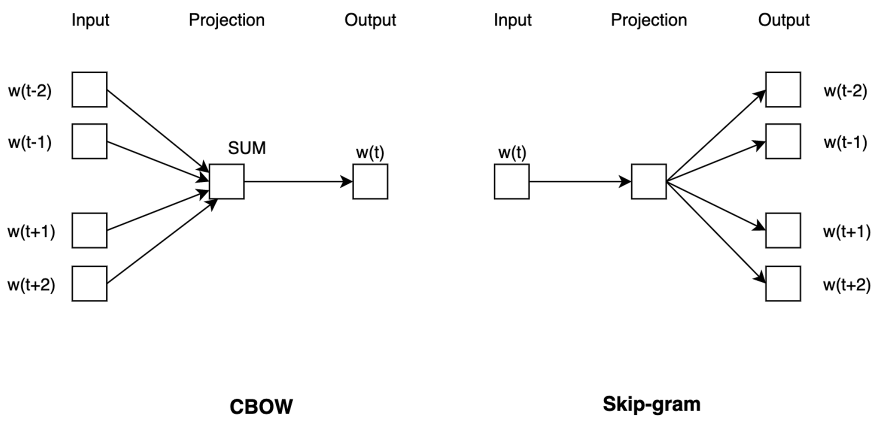

Figure 3.6: Word2vec models, inspired by [34]

3.7.2 Global Vector

Global Vector (GloVe) is an extension of Word2Vec which was developed at Stanford [7]. A disadvantage of Word2Vec is that it will not have enough corpus to learn the representations with a small amount of data. GloVe gives better performance on word embedding as compared to Word2Vec, with different dataset sizes. The GloVe pre-trained vectors are available under multiple versions with the dimension varies from 25 to 300 and the number of tokens is from 6 billion to 840 billion.

3.8 Text Generation Evaluation Metrics

3.8.1 Bi-Lingual Evaluation Understudy (BLEU)

Bi-Lingual Evaluation Understudy (BLEU) is an algorithm to assess the quality of translated or generated text by machine from one natural language to another [35].

BLEU compares a candidate against multiple references using a *precision* modification. For each n-gram in the candidate, BLEU takes its maximum total count, m_{max}, in any of the references. Those m_{max} are then summed over all distinct n-grams in the candidate and then divided by the total number of n-grams in the candidate to get the modified precision. BLEU's output is always between 0 and 1. The higher the score, the better the translation/generation method.

3.8.2 Recall-Oriented Understudy for Gisting Evaluation (ROUGE)

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [36] is a metric generally used to evaluate the generated text summaries. The ROUGE scores range from 0 to 1, with 1 for the perfect match between the hypothesis and the reference. In this thesis, we use ROUGE-1, ROUGE-2, and ROUGE-L scores the same as the baseline [3]. ROUGE-1 and ROUGE-2 compare the overlap of unigram and bigram respectively between the generated headlines and the references, whereas ROUGE-L measures on the longest common subsequence from the sequence pairs.

Chapter 4

Data

This chapter describes the datasets used in the experiments of this thesis and the data-preprocessing steps that have been employed for each dataset.

4.1 The Webis Bias Flipper 2018 Dataset

The *Webis Bias Flipper 2018* (Webis-Bias-Flipper-2018) [8] dataset contains 2,781 events happened from June 1st, 2012 to February 10, 2018 from the website allsides.com. Each event has a title and a summary. It is also listed with the publishers, biases, headlines, and contents from all corresponding news articles. There're 6,458 articles in total come from three different types of bias category: *From the Right*, *From the Left*, and *From the Center*.

	Number of articles	**Percentage**
From the Right	2,542	39.4%
From the Left	2,388	37.1%
From the Center	1,517	23.5%

Table 4.1: Number of articles from each bias

Table 4.1 presents the number of articles from each bias. In the thesis, we only consider *From the Left* and *From the Right* articles in the bias flipping experiments since we want to perform the bias changing into the extreme. Thus, the dataset that is used in the following experiments only has 4,840 news articles.

As the dataset is published under CSV format, we use Pandas [37] as an easy way to load data and perform pre-processing. We also employ *word_tokenize* and *sent_tokenize* modules from Natural Language Toolkit (NLTK) [38] to do the tokenization. The news articles' contents are tokenized into sentences, then those sentences and the news headlines will be processed by applying *word_tokenize*. All punctuation characters are also removed from the processed text. Some outputs are shown in the table 4.2

Original text	**Output**
Trump Hands Out 'Fake News Awards,' Sans the Red Carpet	*Trump Hands Out Fake News Awards Sans the Red Carpet*
Donald Trump Corrects John Kelly: 'The Wall Is the Wall It Has Never Changed or Evolved'	*Donald Trump Corrects John Kelly The Wall Is the Wall It Has Never Changed or Evolved*

Table 4.2: Pre-processed text examples

As we only perform the bias flipping on the sentence level, we create a sentence-based dataset for training the model and generating text. The detailed information of this dataset is presented in Table 4.3. Both the training set and test set require two separated files for different biases:

- *train1.txt*: training set that contains titles and body's sentences from the left-wing news articles.
- *train2.txt*: training set that contains titles and body's sentences from the right-wing news articles.
- *valid1.txt*: test set that contains titles from the left-wing news articles.
- *valid2.txt*: test set that contains titles from the right-wing news articles.

	Number of sentences	**Average length of the sentences**
train1.txt	92,626	24
train2.txt	77,046	22
valid1.txt	2,388	9
valid2.txt	2,542	10

Table 4.3: Details of the Webis-Bias-Flipper-2018 dataset that is used in our approach

4.2 Hyperpartisan News Dataset

This Hyperpartisan News Dataset [9] consists of two different parts. One part is "by-publisher" data containing the articles that are labeled by the overall bias of the source publisher. Those labels are given BuzzFeed journalists or MediaBiasFactCheck.com.

The "bypublisher" dataset has 750,000 news articles which can be considered as a relatively large dataset. 50% of this part is hyperpartisan articles of which half are on the left bias and another half are on the right. This part of the Hyperpartisan News Dataset is split into the training set and validation set with the ratio 80% - 20% respectively.

Table 4.4 shows some statistics regarding the bias in each set of data. There're five bias categories: *left*, *right*, *least*, *left-center* and *right-center*. Similar to the Bias Flipper 2018 dataset, we only consider the *left* and *right* biased news articles when preparing the data for our experiments.

	right	**left**	**least**	**left-center**	**right-center**
Training set	150,000	150,000	187,114	70,053	42,833
Validation set	37,500	37,500	38,296	23,473	13,231

Table 4.4: The statistic in the bias of the articles in "bypublisher" dataset

Because of the relatively huge number of articles in the "bypublisher" data, we only try to focus on the news headlines to perform the bias flipping which means we prepare a dataset of 300,000 titles for the training set and 75,000 titles for the test set in our experiments.

Both "bypublisher" and "byarticle" parts of Hyperpartisan News Dataset are released as XML files. Thus, we need to parse the XML files first to retrieve the article titles and body contents, then we filter them using the *bias* attribute to get left-wing and right-wing articles. After that, we apply the same pre-processing steps as described in the Webis-Bias-Flipper-2018 dataset section 4.1 above. Tables 4.5 shows the information on the number of pre-processed sentences in "bypublisher" and "byarticle" hyperpartisan news dataset that are used in our experiments.

	bypublisher	**byarticle**
train1.txt	149,840	3,874
train2.txt	148,995	5,209
valid1.txt	28,925	238
valid2.txt	35,380	407

Table 4.5: Number of sentences in "bypublisher" and "byarticle" hyperpartisan news dataset

Chapter 5

Solution Approach

This chapter describes the model that is used to perform the experiments in this thesis. We employ ARAE as the base model to do the bias flipping task.

5.1 Existing Approaches/Baselines

An early approach to this problem was proposed in paper *Learning to Flip the Bias of News Headlines* [3]. They flip the bias of headlines using an autoencoder-based network. This bias flipper is trained by the left-wing and right-wing articles. Figure 5.1 illustrates the overview of their work.

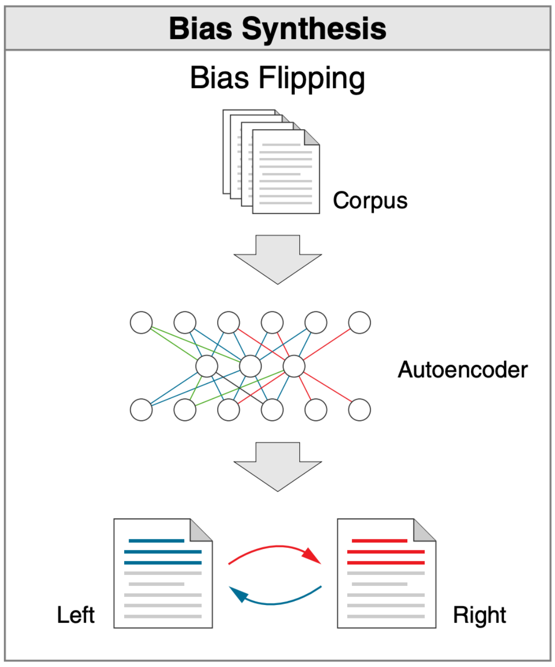

Figure 5.1: Bias flipping overview from [3]

In order to train the flipping model, they build a training set from the content of articles, break them into sentences in particular. On the other hand, the validation and the test set only contain news titles.

They train two different encoders, one for flipping the bias from left to right, and the other one is for the right to left. Let's s_o, s_t denote the original and target sentence; b_o, b_t denote the original and target bias label; and z_o, z_t are the corresponding content. The two encoders are $E(s_k, b_k), k \in \{o, t\}$.

They also train two generators G to generate sentence from given bias and article content:

$$\hat{s}_k \sim G(z_k, b_k) = p(s_k|b_k, z_k) \tag{5.1}$$

The loss function is constructed from the reconstruction error $\mathcal{L}_{rec}(\theta_E, \theta_G)$ and the loss of adversarial discriminator D, $\mathcal{L}_{adv}$, as follows:

$$\mathcal{L}_{rec}(\theta_E, \theta_G) = \mathbb{E}_{s_k \sim S_k}[-\log\, p(s_k|s_k, E(s_{\bar{k}}, b_{\bar{k}}))] \tag{5.2}$$

$$\mathcal{L}_{adv} = -\log\, D_k(s_k) - \mathbb{E}[\log(-D_k(\hat{s}_{\bar{k}}))] \tag{5.3}$$

$$\mathcal{L} = \mathcal{L}_{rec_{o\to t}} + \mathcal{L}_{rec_{o\to t}} - (\mathcal{L}_{adv_{o\to t}} + \mathcal{L}_{adv_{o\to t}}) \tag{5.4}$$

where $\bar{k}$ is o when k is t and the other way round; θ_E and θ_G are the parameters of E and G; D_k is the discriminator.

To reduce the loss, they minimize both the reconstruction error and the loss of the adversarial discriminator. They apply the setup of cross-aligned autoencoder [16] to train the model with a number of sentences from both left and right. They get 41.5% of the generated sentences that still keep the semantics while the sentiment is modified.

They also build a dataset including 2196 headline pairs, each pair consists of one left-biased sentence and one right-biased sentence. To evaluate the generated text from the test set, they hired three journalists to annotate them based on four questions [3]:

Q1. Do you understand headline 1 (the original headline)? Answers: *yes / partially yes / no / not sure*

Q2. Do you understand headline 2 (the generated headline)? Answers: *yes / partially yes / no / not sure*

Q3. Do both headlines report on the same event? Answers: *same / mostly same / changed / not sure*

Q4. Do the headlines have opposite bias? Answers: *flipped / partially flipped / same / not sure*

Out of 200 generated sentences, they got 36.5% as understandable. 63.5% of generated headlines still keep the semantics of the original headline, and 65% of those sentences have the bias flipped. Among the generated sentences with changed content, the changes of bias also appear on 28%. They also perform the automatic evaluation using ROUGE score [36] to calculate the similarity between the original and the generated headlines. The results in F-scores with ROUGE-1, ROUGE-2, ROUGE-L are 15%, 3% and 12% respectively. We will use those results as the baseline to evaluate the outcome of the approach in this thesis.

5.2 Analysis

The work of bias flipping is closely related to text style transfer. In the approach described in section 5.1 above, cross-aligned AE has been leveraged and extended to perform the task of unaligned text style transfer. Regarding this problem, in the paper *Adversarially Regularized Autoencoders* [4] they have compared the performance of the ARAE model and the cross-aligned AE model from [16] on Yelp reviews corpus. They follow the setting of cross-aligned AE and split the dataset into positive and negative reviews. For the quantitative evaluation, they use both automatic methods such as (1) Transfer: how the model change the sentiment of the sentences; (2) BLEU score: the quality of generated text by a machine from one natural language to another; and human assessment by choosing 1000 sentences randomly (50% are positive and 50% are negative) then asking the crowdworkers to assess the sentiment and naturalness of the generated sentences.

Model	**Transfer**	**BLEU**
Cross-Aligned AE	77.1%	17.75
ARAE	81.8%	20.18

Table 5.1: Automatic Evaluation between ARAE and Cross-Aligned AE [4]

Model	**Transfer**	**Naturalness**	**Similarity**
Cross-Aligned AE	57%	2.7	3.8
ARAE	74%	3.8	3.7

Table 5.2: Human Evaluation between ARAE and Cross-Aligned AE [4]

Table 5.1 and table 5.2 report the results of this evaluation. ARAE model clearly shows the potential to be applied for bias flipping problems as the way Cross-aligned AE has been used to improve the performance, although the adversarial regularization model seems to generate text that less similar to the source sentences.

5.3 Proposed Solution

The work from [3] and [2] have shown that the articles' writing style could be used to detect the hyperpartisan news or to transfer the bias from left to right or from right to left.

The approach we propose is based on the ARAE model. This model enhances the adversarial autoencoder (AAE) [39] by learning a parameterized prior as a Generative Adversarial Networks (GAN) [6] and apply to discrete sequences. In the original work, the ARAE model is trained to generate natural text and transfer unaligned textual style. Our idea is to leverage the ARAE model for bias flipping.

ARAE can also be formalized as a latent variable model under the Wasserstein autoencoder (WAE) framework [40]. The model contain a discrete autoencoder regularized with a prior distribution which consists of an encoder $enc_\phi : X \to Z$ and two conditional decoders $p_\psi(\mathbf{x}|\mathbf{z}, y)$, where y is the label and z is the last hidden state of an encoder. The decoders use a softmax layer as the output layer:

$$p_\psi(\mathbf{x}|\mathbf{z}, y) = \prod_{j=1}^{n} softmax(\mathbf{W}h_j + \mathbf{b})_{x_j} \tag{5.5}$$

where $\mathbf{W}$ and $\mathbf{b}$ are the decoder parameters and h_j is the hidden state.

5.3.1 Adversarially Regularized Autoencoder

ARAE is a combination of an autoencoder and GAN-regularized latent representation. The discrete autoencoder is regularized and can be illustrated as follows:

$$\min\nolimits_{\phi,\psi} \mathcal{L}_{rec}(\phi, \psi) + \lambda^{(1)} W(\mathbb{P}_Q, \mathbb{P}_\mathbf{z}) \tag{5.6}$$

with $\mathcal{L}_{rec}(\phi, \psi)$ is cross-entropy reconstruction loss:

$$\mathcal{L}_{rec}(\phi, \psi) = -\log\, p_\psi(\mathrm{x}|\mathrm{enc}_\phi(\mathrm{x})) \tag{5.7}$$

where W is the Wasserstein distance [40] between the distribution of $enc_\phi(x)$, $\mathbb{P}_Q$, and a prior distribution $\mathbb{P}_\mathbf{z}$; ϕ, ψ are the parameters of encoder and decoder respectively and λ is the weight factor.

The model is trained iteratively through five steps [4]:

1. **Train the encoder and decoder to minimize the reconstruction error.** (ϕ, ψ)

Sample from a true distribution $\{\mathbf{x}^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_*$ and compute the last hidden state $\mathbf{z}^{(i)} = enc_\phi(\mathbf{x}^{(i)})$. Backprop loss:

$$\mathcal{L}_{rec} = -\frac{1}{m}\sum_{i=1}^{m} \log p_\psi(\mathbf{x}^{(i)}|\mathbf{z}^{(i)}) \tag{5.8}$$

2. **Train the critic function to approximate the Wasserstein distance W.**

   Sample from a true distribution $\{\mathbf{x}^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_*$ and the noise from a Gaussian distribution $\{\mathbf{s}^{(i)}\}_{i=1}^{m} \sim \mathcal{N}(0, \mathbf{I})$. Compute $\mathbf{z}^{(i)} = enc_\phi(\mathbf{x}^{(i)})$ and the generator function on the noise: $\tilde{\mathbf{z}}^{(i)} = g_\theta(\mathbf{s}^{(i)})$, where θ is the generator's parameters. Backprop loss:

$$\mathcal{L}_{cri} = -\frac{1}{m}\sum_{i=1}^{m} f_w(\mathbf{z}^{(i)}) + \frac{1}{m}\sum_{i=1}^{m} f_w(\tilde{\mathbf{z}}^{(i)}) \tag{5.9}$$

   where f_w is the critic/discriminator function to distinguish the real data and the generated data from the generator g_θ. The critic parameters w is clipped to $[-\epsilon, \epsilon]^d$.

3. **Train a bias attribute classifier** (u)

   The model learns to remove the bias attribute y from the prior, which means all significant information is encoded except information about the bias. With extension from Equation 5.6, the latent space bias attribute classifier is as follows:

$$\min_{\phi,\psi,\theta} \mathcal{L}_{rec}(\phi, \psi) + \lambda^{(1)} W(\mathbb{P}_Q, \mathbb{P}_{\mathbf{z}}) - \lambda^{(2)} \mathcal{L}_{class}(\phi, u) \tag{5.10}$$

   where $\mathcal{L}_{class}(\phi, u)$ is the loss function of the attribute classifier $p_u(y|\mathbf{z})$:

$$\mathcal{L}_{class}(\phi, u) = -\frac{1}{m}\sum_{i=1}^{m} \log p_u(y^{(i)}|\mathbf{z}^{(i)}) \tag{5.11}$$

   where $\mathbf{z}^{(i)} = enc_\phi(\mathbf{x}^{(i)})$ with $\{\mathbf{x}^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_*$ and $y^{(i)}$ is the corresponding bias label.

4. **Train the encoder/generator adversarially (ϕ, θ) to minimize W.**

   Sample from a true distribution $\{\mathbf{x}^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_*$ and the noise from a Gaussian distribution $\{\mathbf{s}^{(i)}\}_{i=1}^{m} \sim \mathcal{N}(0, \mathbf{I})$. Compute $\mathbf{z}^{(i)} = enc_\phi(\mathbf{x}^{(i)})$ and $\tilde{\mathbf{z}}^{(i)} = g_\theta(\mathbf{s}^{(i)})$. Backprop loss:

$$\frac{1}{m}\sum_{i=1}^{m} f_w(\mathbf{z}^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(\tilde{\mathbf{z}}^{(i)}) \tag{5.12}$$

5. **Train the encoder adversarially (ϕ) to minimize the loss of the classifier.**

   Compute $\mathbf{z}^{(i)} = enc_\phi(\mathbf{x}^{(i)})$ where $\{\mathbf{x}^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_*$ and $y^{(i)}$ is the corresponding bias label. Backprop loss:

$$\mathcal{L}_{class}(\phi, u) = -\frac{1}{m}\sum_{i=1}^{m} \log p_u(1 - y^{(i)}|\mathbf{z}^{(i)}) \tag{5.13}$$

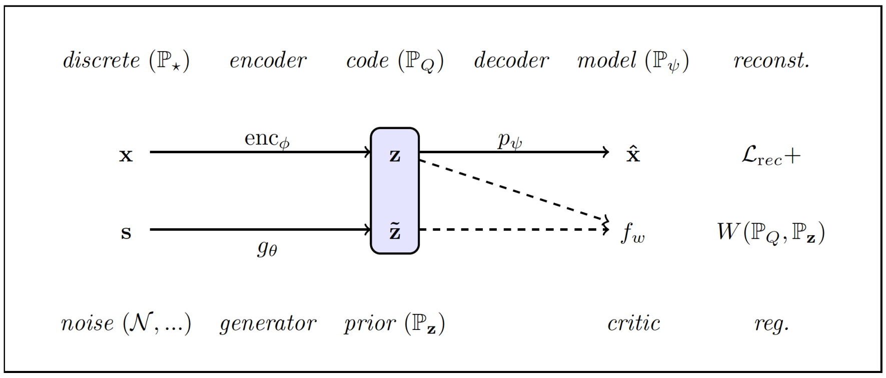

Figure 5.2: ARAE architecture [4]

Figure 5.2 describes the full architecture of the ARAE model.

5.3.2 ARAE for bias flipping

A one-layer LSTM is used for both encoder and decoder. ARAE model is trained with two separate decoders, one for left-biased articles $p_\psi(\mathbf{x}|\mathbf{z}, y = 0)$ and one for right-biased articles $p_\psi(\mathbf{x}|\mathbf{z}, y = 1)$. The encoder is trained adversarially by encoding the titles and sentences from the article body from one class and decoding with the decoder corresponds to the other class.

5.3.3 Vocabulary

We use GloVe [7] pre-trained vectors as word embedding for the neural networks rather than the word indices in the vocabulary as ARAE's implementation [4]. We use the *Wikipedia 2014 + Gigaword 5* version which was trained on a corpus of 6 billion tokens and includes 400 thousand tokens. Each word in the vocabulary is represented by a vector of dimension 300. Both the encoder and decoder also have a hidden layer of 300 units.

The vocabulary size V is a hyper-parameter and can be configured during the training process. In our experiments, we only use the vocabulary size of 30000 tokens. Vocabulary building is as follows:

- Tokenize all the sentences in the corpus
- Count the number of occurrences of each token and sort by token frequency in descending order to make the vocabulary deterministic
- Prune by most frequently seen tokens from the sorted list

- Look up the GloVe vector representation for each token

Following the setting from ARAE, we also add some special tokens into the vocabulary:

- *<BOS>*: *begin-of-sequence* token. The generator can add this token to the beginning of a sentence.
- *<EOS>*: *end-of-sequence* token. The generator will generate this token to indicate a sentence ends.
- *<UNK>*: *unknown-token*. Tokens that are not in the vocabulary will be replaced by this UNK token.
- <PAD>: *padding-token*. When the length of a generated sentence is smaller than the predefined length, the generator will add PAD tokens to the end of the sequence until reaching the predefined length.

The following is an example of generated text with special tokens:

donald trump <unk> dominance leaves gop establishment banking on brokered convention <eos> <pad> <pad> <pad> <pad> <pad>

5.3.4 Implementation

Since the ARAE model was implemented on *Pytorch* framework [41], we also base on this implementation and extend the work using Pytorch. We also use Python for all of the other work in this thesis such as data pre-processing and experiment result evaluation.

To evaluate the quality of the generated text, a Python library for the ROUGE metric has been used to compute ROUGE scores. The model in this thesis is trained on servers with high-performance GPUs. Details of hardware and software that are used for model training are shown in Table 5.3.

The code repository of this thesis is available at

`https://github.com/khoaln/master-thesis-fake-news-detection`.

	Information
OS	Ubuntu 18.04.4 LTS
CPU	Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
GPU	Tesla V100-PCIE-32GB
Cuda	8
Python	3.6.8
PyTorch	0.3.1
NLTK	3.5

Table 5.3: Detailed information on resources that are used to train the model and perform the experiments

Chapter 6

Experimental Evaluation

The experiments are to examine how ARAE, a generative adversarial network combined with discrete autoencoder, performs on bias flipping problems. The experiments are conducted mainly on Webis Bias Flipper 2018 dataset, but there're also some attempts with data from Hyperpartisan News Dataset as well.

6.1 Experimental Setup

6.1.1 Model setup and parameters

The ARAE model consists of 4 modules: an autoencoder that contains one encoder and two different decoders for different biases, a GAN generator model, a GAN discriminator model, and a critic/classifier model. The architecture of those models and the parameters giving the best experimental result on Webis Bias Flipper 2018 dataset are described below.

1. **Autoencoder**:
    - *Word embeddings*: both the encoder and the two decoders have an embedding layer of dimension 300. The number of tokens is 30,004 in which 30,000 from the vocabulary and the other four are special tokens as mentioned in Chapter 5.
    - *Encoder*: is an one-layer LSTM with 128 hidden units. Input size is 300.
    - *Decoder*: is an one-layer LSTM with 128 hidden units. Input size is 428.
    - SGD optimization with learning rate 1
    - Standard deviation of noise: 0.1

- Dropout: 0

2. **Generator**:
   - The generator is an ANN with two 128-units hidden layers, input size is 32 and output size is 128. Denoted as 32-128-128-128
   - Activation function: ReLU
   - Optimized by Adam, learning rate 5e-05
3. **Discriminator**:
   - The discriminator is an ANN 128-128-128-1
   - Activation function: Leaky ReLU, negative slope 0.2
   - Optimized by Adam, learning rate 1e-04
4. **Critic**:
   - The critic is an ANN 128-128-128-1
   - Activation function: ReLU
   - Optimized by Adam, learning rate 1e-05
   - The critic is trained 5 iterations in every loop

The model is trained with a batch size of 128 and 500 epochs. The hyper-parameters tuning is performed on the parameters below based on the reconstruction error as a metric:

- Number of epochs: 100, 200, 300, 400, 500
- Autoencoder learning rate: 0.1, 0.5, 1, 5, 10
- GAN generator learning rate: 5e-04, 5e-05
- Critic learning rate: 1e-05, 5e-05

The experiments on Hyperpartisan News Dataset are performed following the model setup and tuned parameters for the Webis Bias Flipper 2018 dataset. Because of the long training time, we can only experiment with 200 epochs for "byarticle" data and 100 epochs for "bypublisher" data.

6.1.2 Evaluation metrics

In the automatic evaluation, besides the ROUGE scores, we also leverage a simple CNN-based classifier [3] to evaluate if the generated headlines have the same attributes as the original headlines within the same bias. The model is trained by all news body's sentences from left-biased and right-biased articles of the Webis Bias Flipper 2018 dataset, with a batch size of 96, the number of epochs is 100 and optimized by Adam, learning rate 1e-04.

In an additional manual evaluation, the same sentences in the test set of the baselines [3] can be used to generate the bias-flipped headlines. Then, we could hire some journal experts to evaluate the generated text based on three questions (Q2, Q3, Q4) mentioned in Chapter 5. Unfortunately, we couldn't set up this in time for the thesis but it may be useful for future evaluation.

6.2 Experimental Results and Discussions

6.2.1 Webis Bias Flipper 2018

Parameters	Reconstruction error
lr_ae=1,lr_classify=5e-05,lr_gan_g=5e-04	4.03
lr_ae=0.5,lr_classify=5e-05,lr_gan_g=5e-04	4.48
lr_ae=0.1,lr_classify=1e-05,lr_gan_g=5e-05	5.45
lr_ae=1,lr_classify=1e-05,lr_gan_g=5e-05	4.08
lr_ae=10,lr_classify=1e-05,lr_gan_g=5e-05	3.33
lr_ae=5,lr_classify=1e-05,lr_gan_g=5e-05	**3.17**

Table 6.1: Reconstruction error with different learning rates

Table 6.1 shows the reconstruction error when experimenting with different autoencoder learning rate, classifier learning rate, and the GAN generator learning rate. These experiments are performed to examine which set of learning rates that can achieve the lowest construction error after 100 epochs. From the results, we can see that (lr_ae=5, lr_classify=1e-05, lr_gan_g=5e-05) gives the best result. The learning rates on the critic and the generator are similar to the setting used in the experiments of text style transfer from ARAE paper [4]. Each experiment of those consumes 17 hours for training the model in 100 epochs.

From that result, the experiments are continuously conducted with multiple numbers of epochs from 100 to 500 in order to examine the changes of loss and evaluate the quality of generated headlines at different epochs. The result of loss values is reported in

Table 6.2. We can see that the loss is reduced with more epochs. Due to the limit of

Epochs	100	200	300	400	450	500
Rec. error	3.17	2.52	2.14	1.81	1.68	1.57

Table 6.2: Reconstruction error at different epochs

time and resources on the scope of this thesis, we don't increase the number of epochs to check if the reconstruction error will be more decreased. We leave this for future work.

We also examine the quality of the generated sentences from each experiment. In order to do this, one original headline from left-wing and one from the right-wing has been chosen to evaluate the corresponding generated headlines.

Original Headline	the 'muslim ban' president is about to give a speech on islam in saudi arabia
epoch 100th	the speech is a free speech to make america on about muslims in the world
epoch 200th	the president is not a muslim speech to address islam in saudi arabia
epoch 300th	the president is about to make a muslim ban on muslim countries in an era
epoch 400th	the president is calling on muslim countries to build a muslim ban in an effort to address the muslim world
epoch 500th	the president is about muslim countries in a speech to the muslim world

Table 6.3: Generated left-to-right headlines at different epochs

Original Headline	house republicans under pressure by trump to deliver compromise bill for obamacare repeal
epoch 100th	gop senators who voted for trump to repeal obamacare
epoch 200th	house gop senate bill would require by republicans to repeal obamacare by changing obamacare
epoch 300th	house gop senate bill would allow them to keep insurance for coverage and tax cuts to keep insurance through
epoch 400th	house republicans would try to repeal obamacare even if they fail to keep him for votes
epoch 500th	house republicans would try to repeal obamacare as much as bill passes than ever vote

Table 6.4: Generated right-to-left headlines at different epochs

Table 6.3 and 6.4 show examples of generated text between the opposite biases. The blue color indicates parts of the generated sentences that try to keep the semantics (same event) as of the original headlines including the entities and phrases. The red color mentions parts of the generated headlines that try to flip the original bias. Those examples also show that the higher number of training epochs, the more meaningful generated sentences are. It also tries to keep the generated headlines closer to the original semantics but with flipped bias.

In order to perform the automatic and human evaluation and compare it with the result of the baselines [3], we try to flip the bias of the same 100 headlines of the baselines test. It includes 50 sentences from the left and the same number of sentences from the right.

We calculate again the ROUGE F1 scores on the generated headlines from both models. The results are reported in Table 6.5. The ROUGE score is insufficient to evaluate bias

	ROUGE-1	ROUGE-2	ROUGE-L
Baseline	0.24	0.06	0.20
ARAE	**0.28**	**0.07**	**0.24**

Table 6.5: Compare ROUGE in F1 scores with the baselines [3]

flipping quality [3], but it can be used to assess the ability to keep on the same events of the generated text.

	Accuracy
Original headlines	59%
Baseline	51.5%
ARAE	**56.5%**

Table 6.6: Accuracy of sentence-level bias classification on different data from the simple classifier [3]

Table 6.6 reports the bias-classification accuracy using a simple CNN model. The model is still far to become a highly effective bias classifier, but it can be used as a reference to examine that the bias attribute in the generated text by the ARAE model is closer to the bias of original headlines than the baselines.

6.2.2 Hyperpartisan News

We use the same settings and parameters that give the best result on Webis Bias Flipper 2018 dataset but with fewer epochs. The experiments on a small dataset, "byarticle", show that a generative adversarial network clearly need more data to train and generate meaningful sentences. With 50 epochs, the model only generates *"the the the the ..."* for every original headline input. When the number of epochs is increased to 100, the generated text quality isn't improved much. It contains many repeated phrases such as:

"the fbi has been been in the united states of the united states states the united states states the united states states the united states states the united states states the united states"

"the fbi s campaign is the president trump s campaign and the president 's campaign and the president 's campaign"

The results are slightly better with 200 epochs. It's able to generate different sentences for different input and fewer repeated words. The meaning of the generated text and bias flipping is still not good:

"trump also called a statement on friday night that he had been a muslim white house and the white house"

"trump s comments about his wife s story"

"police say they were found on the news conference"

However, when the number of epochs is increased to 300 or more, the model only generates one output for every original headline:

"the associated press contributed to this report"

It's apparently difficult to train this model with a limited amount of data like the "byarticle" dataset. The training time on this dataset is also very short, only 2 hours 20 minutes for 500 epochs.

Training on the bigger dataset, "bypublisher", takes a lot more time. It requires 22 hours for 100 epochs. We can only perform two experiments with 100 epochs of training and different learning rate.

Original and generated headlines	**Bias**
agricultural group seeks to employ veterans *environmental seeks to teach immigrants*	left *right*
where have all the rock stars gone *where we all the stars clash holds off spring*	left *right*
get ready for spectre the next bond movie *get ready for the next bond film*	left *right*
coal industry suggests more mountaintop mining bush appointee just walks out *coal industry rules created to amend mine spill in*	right *left*
democrats seek to fix bug in obamacare *democrats seek to fix obamacare in nuke*	right *left*

Table 6.7: Generated headlines on "publisher" dataset after 100 epochs

The examples from Table 6.7 show that the model can give a promising output with just 100 epochs when being trained with enough amount of data. The bias of the input headlines in "bypublisher" dataset is labeled by the overall bias of the publisher can also be a part of the contribution to the result as a publisher often has their own writing style and it’s different from each other. However, the model will need to be trained with more epochs or other fine-tuned parameters in future work as the generated text still doesn’t satisfy the semantic requirements and bias flipping.

Chapter 7

Conclusion and Future Directions

7.1 Conclusion

In this thesis, we have examined the abilities to flip the bias of news articles headline using an adversarial model on several datasets. Our experiments indicate that, although we achieve a slightly better result than the baselines in some metrics, the generative adversarial network needs more effort to solve the task of bias flipping but still having the content kept. Especially with the headline level as bias can appear at the article or paragraph levels only.

Even though the results still need more improvements, ARAE shows potential in applying adversarially trained models for bias flipping tasks with room for further research. The next section will suggest some directions to study more in the future to improve the model as well as extend the bias flipping problem.

7.2 Future Directions

7.2.1 Perform more experiments on hyperparameters tuning

As mentioned in Chapter 6, we can only execute the experiments with some set of parameters based on the settings from ARAE paper due to the time constraints. During the tuning process, we also use reconstruction error as the only target to optimize. Involving other loss values could help to find a better configuration and may also reduce the training time. We could try a higher number of epochs as well.

7.2.2 Train with other datasets

The bias attribute of the news articles, especially the political articles, may have some properties in common regardless of the source of publishing, author, or writing styles. Training the model on multiple bias datasets can help to study more on this attribute. That could be useful for research on an effective biased text generator or classifier.

7.2.3 Leverage the critic of the ARAE model to use as a Political Bias Classifier

While working on bias flipping using ARAE, we see that the module *critic* has the potential to become a bias classifier since it is used to distinguish real data and generated text. We have tried the critic in some classification experiments but that's not successful. More research is needed in this aspect.

7.2.4 Automatic evaluation metrics

During this thesis and also from the conclusion from the baseline [3], the current metrics for bias flipping problems seem to not complete and insufficient. Therefore, developing a sophisticated evaluation metric is needed in this situation and its study is as important as a bias-oriented generation or classification study.

List of Figures

3.1 A simple ANN 6
3.2 Illustration of a RNN 7
3.3 Illustration of a LSTM unit 8
3.4 Simple Autoencoder architecture 9
3.5 GAN general architecture illustration, inspired by [30] 10
3.6 Word2vec models, inspired by [34] 11

5.1 Bias flipping overview from [3] 17
5.2 ARAE architecture [4] 22

List of Tables

4.1 Number of articles from each bias 13
4.2 Pre-processed text examples 14
4.3 Details of the Webis-Bias-Flipper-2018 dataset that is used in our approach 14
4.4 The statistic in the bias of the articles in "bypublisher" dataset 15
4.5 Number of sentences in "bypublisher" and "byarticle" hyperpartisan news dataset 15

5.1 Automatic Evaluation between ARAE and Cross-Aligned AE [4] 19
5.2 Human Evaluation between ARAE and Cross-Aligned AE [4] 19
5.3 Detailed information on resources that are used to train the model and perform the experiments 24

6.1 Reconstruction error with different learning rates 27
6.2 Reconstruction error at different epochs 28
6.3 Generated left-to-right headlines at different epochs 28
6.4 Generated right-to-left headlines at different epochs 28
6.5 Compare ROUGE in F1 scores with the baselines [3] 29
6.6 Accuracy of sentence-level bias classification on different data from the simple classifier [3] 29
6.7 Generated headlines on "publisher" dataset after 100 epochs 30

Bibliography

[1] Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*, 2017.

[2] Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. A stylometric inquiry into hyperpartisan and fake news. In *ACL*, volume 1, pages 231–240, 2018.

[3] Wei-Fan Chen, Henning Wachsmuth, Khalid Al-Khatib, and Benno Stein. Learning to flip the bias of news headlines. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 79–88, Tilburg University, The Netherlands, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6509. URL https://www.aclweb.org/anthology/W18-6509.

[4] Junbo, Zhao, Y. Kim, K. Zhang, A. M. Rush, and Y. LeCun. Adversarially Regularized Autoencoders for Generating Discrete Structures. *ArXiv e-prints*, June 2017.

[5] Ilya O. Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein auto-encoders. *ArXiv*, abs/1711.01558, 2018.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[7] Jeffrey Pennington, Richard Socher, and Christoper Manning. Glove: Global vectors for word representation. volume 14, pages 1532–1543, 01 2014. doi: 10.3115/v1/D14-1162.

[8] Johannes Kiesel, Florian Kneist, Milad Alshomary, Benno Stein, Matthias Hagen, and Martin Potthast. Reproducible Web Corpora: Interactive Archiving with Automatic Quality Assessment. *Journal of Data and Information Quality (JDIQ)*,

10(4):17:1–17:25, October 2018. doi: 10.1145/3239574. URL https://dl.acm.org/doi/10.1145/3239574.

[9] Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, David Corney, Payam Adineh, Benno Stein, and Martin Potthast. Data for PAN at SemEval 2019 Task 4: Hyperpartisan News Detection, November 2018. URL https://doi.org/10.5281/zenodo.1489920.

[10] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018.

[11] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. *ArXiv*, abs/1709.08624, 2018.

[12] Md. Akmal Haidar and Mehdi Rezagholizadeh. Textkd-gan: Text generation using knowledge distillation and generative adversarial networks. *Lecture Notes in Computer Science*, page 107–118, 2019. ISSN 1611-3349. doi: 10.1007/978-3-030-18305-9_9. URL http://dx.doi.org/10.1007/978-3-030-18305-9_9.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[14] Joosung Lee. Stable style transformer: Delete and generate approach with encoder-decoder for text style transfer, 2020.

[15] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward controlled generation of text. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/hu17e.html.

[16] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6830–6841. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7259-style-transfer-from-non-parallel-text-by-cross-alignment.pdf.

[17] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. doi: 10.1007/BF02478259. URL https://doi.org/10.1007/BF02478259.

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0. URL http://www.nature.com/articles/323533a0.

[19] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In José Mira and Francisco Sandoval, editors, *From Natural to Artificial Neural Computation*, pages 195–201, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. ISBN 978-3-540-49288-7.

[20] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL http://proceedings.mlr.press/v15/glorot11a.html.

[21] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015.

[22] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL http://arxiv.org/abs/1412.6980. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[24] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

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

[26] Nikhil Buduma and Nicholas Locascio. *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. O'Reilly Media, Inc., 1st edition, 2017. ISBN 1491925612.

[27] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1181. URL https://www.aclweb.org/anthology/D14-1181.

[28] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. page 233–243, 1991.

[29] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. ISSN 1935-8245. doi: 10.1561/2200000056. URL http://dx.doi.org/10.1561/2200000056.

[30] Johan Björk and Karl Svensson. Abstractive document summarisation using generative adversarial networks, 2018. URL https://hdl.handle.net/20.500.12380/255471.

[31] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. 12 2016.

[32] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 2852–2858. AAAI Press, 2017. URL http://dl.acm.org/citation.cfm?id=3298483.3298649.

[33] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL http://arxiv.org/abs/1301.3781.

[34] Giuseppe Futia, Antonio Vetro, Alessio Melandri, and Juan Carlos De Martin. Training neural language models with sparql queries for semi-automatic semantic mapping. *Procedia Computer Science*, 137:187–198, 01 2018. doi: 10.1016/j.procs.2018.09.018.

[35] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.

[36] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. page 10, 01 2004.

[37] The pandas development team. pandas-dev/pandas: Pandas, February 2020. URL https://doi.org/10.5281/zenodo.3509134.

[38] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[39] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. 11 2015.

[40] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf. Wasserstein auto-encoders. In *6th International Conference on Learning Representations (ICLR)*, May 2018. URL https://openreview.net/forum?id=HkL7n1-0b.

[41] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.