# Towards an improved understanding of the concept of style and its implications for textual style transfer

Somayeh Jafaritazehjani

2023

# Towards an improved understanding of the concept of style and its implications for textual style transfer

Somayeh Jafaritazehjani
*Technological University Dublin*, d18128486@mytudublin.ie

# Towards an improved understanding of the concept of style and its implications for textual style transfer


**Somayeh Jafaritazehjani**

Supervisor: Prof. John D. Kelleher (TU Dublin)

Dr. habil. Gwénolé Lecorvé (Orange)

Prof. Damien Lolive (University of Rennes 1)

School of Computer Science

Technological University Dublin

A Thesis Submitted for the degree of DOCTOR OF PHILOSOPHY (Ph.D.)

December 2023

I would like to use this opportunity to show my gratitude to those who supported, helped and encouraged me throughout the time I was doing my thesis project.

First and foremost, I would like to appreciate my supervisor Prof. John D. Kelleher for his guidance, assistance and involvement in every step throughout the process. I consider myself truly lucky that I had the opportunity to work with him.

I would also like to thank my supervisors Dr. Gwenole Lecorve and Prof. Damien Lolive and all my colleagues and friends in Expression group in France for their help in my research, fruitful discussions and their support.

Finally, I would like to express my deepest appreciations to my husband, Fredrik, whose constant support encouraged me to keep going and dedicate this thesis to him, and our little Darien.

# Declaration

I certify that this report which I now submit towards the confirmation examination, is entirely my own work and has not been taken from the work of others, and to the extent that such work has been cited and acknowledged within the text of my work.

This report was prepared according to the regulations for graduate study by research of the Technological University Dublin (TU Dublin) and has not been submitted in whole or in part for another award in any other third level institution.

The work reported in this report conforms to the principles and requirements of the TU Dublin's guidelines for ethics in research.

TU Dublin has permission to keep, lend or copy this report in whole or in part, on condition that any such use of the material of the report be duly acknowledged.

Signature:———————————

Date:———————————-

# Acknowledgements

This research has been realized under the ANR (French National Research Agency) projects TREMoLo (ANR-16-CE23-0019) and TextToKids (AAPG 2019).

This project has also been supported by the ADAPT Centre for Digital Content Technology which is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

# Abstract

The concept of linguistic style denotes that many aspects of text can vary while maintaining a same source core semantic meaning. For example, a message may be written in a formal or informal style. The textual style transfer problem aims at generating a paraphrase of a given text by modifying its style while preserving its content. To the best of our knowledge, within the literature on textual style transfer, there is no standard widely accepted definition of the concept of style. Moreover, very few works have investigated the characteristics of language styles. Therefore, previous research, as far as our knowledge extends, have not taken the variations of different textual style domains into account while dealing with the style transfer task. This research investigates domain-specific style characteristics by examining the separation of the style and content, as well as the variations in style across domains and how these variations are encoded. Furthermore, it looks into the factors which are relevant to do a comprehensive evaluation of textual style transfer models. The research uses the domains of sentiment and formality.

A variety of frameworks have been employed as textual style transfer models throughout the experiments of the current work where networks such as RNNs, and transformers are used as the encoders, decoders and discriminators. These models are trained in an unsupervised manner, i.e. the data is or is considered as non-parallel. The experimental methodology frames style transfer as a multi-objective problem, evaluating each approach through three aspects of the generated style-shifted outputs: presence of the target style (style-shift power of the approach), presence of the input content (content preservation power of the approach) and

fluency and grammatical correctness of the output sequences. To evaluate these dimensions, various automatic methods are applied which are further confirmed by conducting human evaluation tests. The performance of the style transfer systems reveals a trade-off between the evaluation aspects. This confirms the need of applying this comprehensive evaluation methodology and questions the approach taken in some previous researches where the focus is on one or two evaluation dimensions which can lead to neglecting the disregarded aspect(s).

Our research firstly looks into the separation of style and content in chapters 4 and 5. To do so, different experiments are conducted to probe the latent space of a variety of adversarial RNN-based style-shift frameworks while considering sentiment and formality as the style domains. The main focus of these experiments is to investigate the presence of the source stylistic features, i.e. to analyse how these models encode style-related features in their latent spaces. The results which hold for the two style domains indicate that style cannot be totally separated from content.

A series of experiments are then designed in chapter 5 to examine if the concept of style is consistent across different style domains. This includes experiments which focus on studying the correlation of style and content across the domains, as well as, studying the effect of modifying the latent space across different style domains. The findings indicate that in the case of sentiment there is a closer entanglement between style and content, as compared with formality domain where these elements are less entangled. Observing that the concept of style can vary across different domains shifts the attention of this study towards analysing how this variation is encoded.

To explore the variations across the style domains, a number of experiments are conducted in chapter 6. This includes a series of probing classification tasks are performed to examine how different layers of encoders of adversarial transformer-based style transfer models encode the style of the input. Furthermore, some unigram-based experiments are conducted which further confirm the variations observed. The results indicate that formality is more

globally encoded compared to the sentiment which is more locally encoded. Finally, a series of experiments look into the effect of emphasizing more on encoding the input on the style-shift power of the models across different style domains which is in line with previous results and implies that formality is a more complex style domain to be dealt with in style transfer scope as compared with sentiment. The findings of these experiments contribute to a better understanding of the style and highlight the question of how the characteristics of various style domains can affect framing the textual style transfer task.

This open question is investigated as a final step by conducting some experiments in chapter 6 to illustrate how style characteristics of different styles should be considered when selecting evaluation methods. In particular, it focuses on the content preservation dimension and shows how it can be computed more effectively by considering the variation of the characteristics and encoding of style across the formality and sentiment domains.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Textual style transfer (TST) is a field of natural language processing (NLP) which focuses on reshaping a text so that it shifts to a different style while preserving its content. Similar to many other Natural Language Generation (NLG) problems, TST is expected to generate grammatically correct sequences. In other words, the main objective of this multi-dimensional scope is the creation of style-shifted text comparable to the text generated by humans. However, this has not yet been achieved. The description of the TST draws the attention to the textual style as one of the key elements of this task which, to the extent of our knowledge, has not yet been rigorously defined in the field of TST (Tikhonov et al., 2020; Tikhonov and Yamshchikov, 2018; Jin et al., 2022). Moreover, there are open questions with regards to this concept such as whether style definition varies across different style domains and if so what are the implications of these variations for TST.

This chapter overviews my PhD thesis by firstly discussing the key concepts of TST as well as some of the applications of this filed. Then, it proceeds by explaining the direction, objectives and contributions of my research . Finally, it provides the outline of the thesis.

## 1.1 Textual style transfer

Style is an important component of language which allows the same semantic information to be expressed in different forms. Each of these forms conveys some extra pieces of information, referred to as stylistic features. For instance, the same topic and content shaped in two styles can show different opinions about the topic, different levels of expertise of the author or speaker, different social relations between the participants of the interaction (the author or speaker on one end and the reader or interlocutor on the other side), etc. In fact, the linguistic concept of style can be summarized as the manner of expressing content which is highly dependent to the creators of the text and their choices (McDonald and Pustejovsky, 1985).

The task of TST is typically framed as assuming a domain of style $S$ containing two or more styles ($s_1 \in S$ and $s_2 \in S$) and involves rewriting given a text in style $s_1$ in a desired style $s_2$ while maintaining the content of the text. For example, a domain of style might be $S = \{s_1 = formal, s_2 = informal\}$ and the TST system is tasked with translating text from formal to informal (and vice versa) while maintaining the content. However, in spite of the growing interest in TST, what style entails is still an open question.

**Key concepts in TST** Previous TST research has provided some definitions for the concept of textual style. Some approaches considered style as a holistic concept which is an implicit and integral component of a language. Taking this view, style cannot be explicitly described and each style can be considered as one separate language. This understanding of style is very different from the concept of style underpinning some previous TST work where stylistic features are detected and removed as a preliminary step (Li et al., 2018a; Madaan et al., 2020; Leeftink and Spanakis, 2019; Xu et al., 2018; Zhang et al., 2018a; Sudhakar et al., 2019). The assumption of the latter TST approaches is that style is encoded as a set of discrete explicit linguistic elements like specific words, or markers. These explicit elements

Table 1.1 Some examples of sequences from the styles of formal versus informal

| **Content/ Style** | **Same** | **Different** |
|---|---|---|
| **Same** | I am very grateful to you. | It has been a wonderful evening. |
| **Different** | Thanks very much! | what's up buddy! |

can be identified by measuring relative frequencies in contrast with texts of other styles, or even directly thanks to hand-crafted knowledge (Tikhonov and Yamshchikov, 2018; Jin et al., 2022). This contradiction highlights the importance of clarifying the concept of style since its understanding can inform the adoption of the methodologies while dealing with the TST problem.

Textual style has also been defined based on discrimination, i.e. considering at least two different texts, style can be taken as the consistent variation between the textual data under study. This data-driven definition of style, as opposed to its linguistic understanding enables the broadening of style to include aspects related to the content or topic as acceptable style domains (Jin et al., 2022), for instance, opinion polarity or sentiment. This approach, however, provides a general concept of style which does not discuss the style characteristics in a detailed manner. Each of these available approaches have limitations, for instance, to the extent of our knowledge, neither of them considered the variations between different style domains while providing a definition for style.

Adopting a discrimination/data-driven definition of style presupposes that style is a concept that can be learnt from data, and it is this perspective that informs the approach to style taken in this thesis. Machine learning is the sub-field of artificial intelligence focused on the development and evaluation of algorithms to learn from data (examples), and two of the most popular forms of machine learning are supervised and unsupervised learning (Kelleher, 2019). The distinction between these two types of learning is mainly based on the kind of data that is used for training. In supervised learning, the training of the model is done using a labelled dataset (i.e., each example in the dataset is labelled with the target

output the model is learning to predict) and the model learn by making predictions and doing weight updates based on the error between the prediction and the gold-stand labels in an iterative manner until convergence, i.e., when the model learns to make appropriately good predictions. In unsupervised strategies, on the other hand, training relies on analysing unlabelled data, i.e. discovering the similarities and differences between the samples in the data in order to extract useful features and structures from which the models can learn. Unlabelled data refers to a collection of samples which do not have a desired label, correct answer or ground truth sample. Supervised learning approaches often result in more accurate models as compared with unsupervised techniques. However, a major difficulty with using supervised learning is that it requires that all the examples are labelled and this typically requires human (expert) annotators, which makes the creation of a labelled dataset time-consuming and expensive. This problem is particularly difficult when the human annotators are asked to perform relatively complex tasks such as paraphrasing a text in a specific style. In order to avoid the difficulties posed in preparing parallel training data for textual-style transfer the work in this thesis is focused on unsupervised learning.

Table 1.1 represents some sentence pairs with the style domain *formality* (formal or informal styles). Sequences in row 1 are formal in style but they differ in content and meaning. In column 1, on the other hand, sequences are similar in content but have different styles, formal, versus informal. These sequence pairs illustrate that text is composed of two components, *style* and *content*, which are the main focus of the textual style transfer task. Some previous works on TST have assumed that these two elements are separable. They mostly applied adversarial end-to-end approaches and focused on disentangling style and content as the key step to enable shifting the textual style (Xu et al., 2018; Jin et al., 2022; Yamshchikov et al., 2019; Rabinovich et al., 2017; Dai et al., 2019a; Hu et al., 2017b; Shen et al., 2017; Fu et al., 2018a; John et al., 2019; Romanov et al., 2019; Tian et al., 2018). This raises this question whether style and content are two independent textual components

or entangled elements. To the best of our knowledge, very little work has investigated the style-content separation in the context of TST. The current work, as a step forward towards understanding the characteristics of style, examines the disentanglement of style and content and compares it across the different style domains of *sentiment* and *formality*. Even though there has been controversies in previous researches such as (Zhang et al., 2018a; Tikhonov and Yamshchikov, 2018; Yamshchikov et al., 2019) to consider sentiment as a style, we mainly focused on these two domains to base our analysis on since they are so central to current work on TST[1].

TST is a multi-objective problem which considers shifting the style, preserving the content and generating fluent text. However, in practice, many state-of-the-art papers on TST such as (Fu et al., 2018a; Gröndahl and Asokan, 2020; Hu et al., 2017b; Li et al., 2018b, 2020; Madaan et al., 2020; Xu et al., 2018) do not consider all three of these evaluation dimensions. Consequently, papers do not fully validate their approach and hence, they cannot easily be compared. Moreover, a trade-off between the three aspects of evaluation has been reported in some previous research (John et al., 2019; Li et al., 2020; Tikhonov et al., 2020). This highlights the importance of considering all the three aspects, since considering one or some aspects while disregarding the other(s) can lead to sacrificing the disregarded aspect. Throughout this study, we employ a comprehensive evaluation methodology considering these three evaluation dimensions.

Investigating TST and improving the task of shifting different textual styles is significant since the advancement in this field can make several contributions to the domain of NLP. In fact, NLP researchers have studied this field for a long time in the form of tasks such as summarization and simplification where the textual styles which have been modified are

[1]We observed that in recent years 31 out of 39 TST papers that we reviewed studied the sentiment-shift task (Xu et al., 2018; Zhang et al., 2018a; Sudhakar et al., 2019; Romanov et al., 2019; Singh and Palod, 2018; Shen et al., 2017; John et al., 2019; Hu et al., 2017a; Fu et al., 2018b; Cao et al., 2020; Xu et al., 2019; Jafaritazehjani et al., 2020, 2021; Dai et al., 2019b; Leeftink and Spanakis, 2019; Jin et al., 2019; John et al., 2019; Leeftink and Spanakis, 2019; Li et al., 2018b, 2020; Prabhumoye et al., 2018; Tikhonov et al., 2020; Zhang et al., 2018b) and 5 focused on the formality-shift problem (Cao et al., 2020; Xu et al., 2019; Jafaritazehjani et al., 2021; Rao and Tetreault, 2018; Jin et al., 2019; Niu et al., 2018; Zhang et al., 2020).

"simplicity" and "verbosity", respectively. We describe some of the applications in the next section.

**Some applications of TST** One of the main target applications of rephrasing messages to have a specific style is to facilitate interaction so as to avoid misunderstanding between the participants of a communication. The modification of the texts are done with regards to the recipient(s) of the communication. The interactions can be human-human as well as human-machine. In the other words, rephrasing messages can improve language understanding between humans or with a machine.

Contributing to the field of NLG by improving the ability of the systems to generate text in a desired style can help improve other NLG tasks. Firstly, the strategies used in TST can be applied in other NLG problems such as paraphrase generation and machine translation (MT). Also, using multi-task strategies to frame TST together with other tasks has proved effective (Niu et al., 2018; Zhang et al., 2020) in different scenarios. For instance, combining MT with formality-shift has led to improvements in terms of BLEU score which represents the similarity between the generated outputs and gold reference(s) for these outputs (Niu et al., 2018). Style-specific MT can also be preferable to MT when the translated text is targeting a specific group such as children where MT can be framed together with simplicity-transfer task.

The improvement on the TST task highlights the importance of considering possible risks of applying this technology and raising concern against abusing it. For instance, in some style domains such as sentiment, TST systems can be employed to manipulate online customer reviews such as restaurant, hotel or product reviews towards benefiting the business owners and service providers. Also, stylistic similarities of texts created by different authors (author-specific styles and writing patterns) can be used maliciously to do author profiling. Risks similar to these examples highlights the need for a global reflection on applying and improving techniques to address these issues. For instance, anonymization which is a TST

application focusing on neutralizing the texts from the author-specific styles (Reddy and Knight, 2016; Gröndahl and Asokan, 2020) can help protect the identity of users and alleviate issues relating to the increasing privacy concerns.

Style transfer techniques have been employed to augment data for images processing tasks (Zheng et al., 2019; Jackson et al., 2019). In a simple way, TST can be used as a data augmentation method. To clarify more, given some text, TST frameworks generate similar texts having the same content but different style. This makes the TST techniques suitable to be applied to create data similar to some existing training data for different NLG tasks, namely, paraphrase generation, MT, question answering or summarization.

## 1.2 Research questions and proposed research

Our research aims at answering different questions listed in this section. The primary focus of our work is exploring style characteristics while dealing with TST (questions 1, 2 and 3). It then proceeds by investigating the interaction between these characteristics and TST (question 4). Finally, it looks into the aspects of a comprehensive evaluation for TST models (question 5).

**Question 1: Are style and content separable?**

So far, one of the main objectives of the previous unsupervised TST systems was to disentangle style from the content and create a style-free latent space for inputs. This is mainly based on the presumption that this disentanglement is doable (Xu et al., 2018; Jin et al., 2022; Yamshchikov et al., 2019; Rabinovich et al., 2017; Dai et al., 2019a; Hu et al., 2017b; Shen et al., 2017; Fu et al., 2018a; John et al., 2019; Romanov et al., 2019; Tian et al., 2018). However, to the extent of our knowledge, the possibility of this disentanglement has not been thoroughly investigated the previous research. The preliminary direction of the current

work is style-content separation, since it can lead to a deeper understanding of style and its characteristics. This research question is addressed in chapters 4 and 5.

**Question 2: Is style consistent across domains?**

A necessary direction to further explore the characteristics of style is to investigate the consistency of this concept across various style domains. This research question is addressed in chapter 5.

**Question 3: How does the encoding of style vary across different style domains?**

This research question which is addressed in chapter 6 studies how stylistic features across different style domains are encoded by TST frameworks.

**Question 4: How do the characteristics of style and the task of TST interact?**

This research question, addressed in chapter 6, examines whether extending the knowledge of the concept of style (findings from the research questions 1, 2 and 3) can contribute to TST. In particular, it investigates the implications of the variations of the encoding of style across different domains on the choice of evaluation methods for TST in a given domain.

**Question 5: What factors are relevant for the evaluation of a TST system?**

This research question looks at different evaluation aspects while framing TST which is a multi-dimensional task to introduce a comprehensive evaluation methodology. The current question is addressed in chapters 4, 5 and 6.

## 1.3 Contributions

A variety of contributions have been made on different aspects of the TST task during the course of this research. This section lists a number of these contributions which address the

main research questions listed in section 1.2. These contributions will be discussed in more details throughout this thesis.

1. We find that style and content cannot be totally disentangled. This observation holds for both domains of sentiment and formality.

   This finding arises from the research question 1 by studying the latent space of several TST systems in chapters 4 and 5. To explore the latent space of the frameworks, we performed experiments to investigate the presence of the style of the input sequences in their corresponding latent representations.

2. We find that style is not consistent across different domains, i.e. the concept of style as sentiment is different from the concept of style as formality.

   To investigate the consistency of the style concept across the style domains (research question 2), we use formality and sentiment as a case study. In chapter 5, we applied several TST systems and for each style domain and each TST model, conducted some experiments on the latent space of the systems. Then, we computed the relation between the content preservation power of the model and the presence of the source style in the latent vectors corresponding to the input texts. Doing this enabled us to compare the level of entanglement of the style and content across different style domains. We also did experiments to modify the latent space of the TST frameworks and compare the changes across the style domains.

3. Our findings suggest that there are variations in how style is encoded across different style domains. In particular, in the sentiment domain, style is encoded relatively locally as compared to the formality domain where the style is more globally encoded.

   The related experiments conducted in chapter 6 investigates different layers of encoders of adversarial transformer-based TST models to examine how each layer encodes style across different style domains. The variations observed in the results of this

experiment was further validated by doing a number of unigram-based analysis tasks. Finally, we examined how putting more emphasis on encoding the input can affect the performance of the models. This resulted in TST systems with weaker ability in shifting the formality as compared to shifting sentiment. This observation which addresses research question 3 suggests that formality-shift is a more complex task which is inline with the observation that indicates that formality is more globally encoded compared to sentiment.

4. We find that the selection of the metrics used for content preservation in TST is sensitive to the type of style being transferred. Specifically, we find that the SBERT-based content preservation metrics work better than the Glove-based metrics for formality, and Glove-based metrics do a better job in sentiment domain. We attribute this difference to the domain-specific characteristics of each style and characteristics of each pre-trained embedding model.

   To explore this interaction between style characteristics and TST, we conducted experiments in chapter 6 which indicate that domain-specific characteristics of style can inform the choice of TST evaluation methodologies (research question 4).

5. We propose a multi-factor evaluation framework covering style-shift power, content preservation and fluency which addresses research question 5.

   We find support for this multi-factor methodology through the trade-offs that we consistently observe throughout our experiments in the chapters 4, 5 and 6. The metrics applied in the proposed comprehensive evaluation methodology are further confirmed by using some strategies such as human evaluation.

## 1.4 Notations

Table 1.2 lists all the notations used throughout this report.

Table 1.2 Notations used throughout the thesis report.

| Notations | Description |
|---|---|
| $\mathbf{x}$ | A sequence of length $T$ with tokens of $[w_1, ..., w_T]$ and unknown style |
| $\mathbf{x}_s$ | A sequence of length $T$ with tokens of $[w_1, ..., w_T]$ and style $s$ |
| $\mathbf{z}$ | The latent representation of $\mathbf{x}$ generated by the encoder of a TST model |
| $\mathbf{z}_s$ | The latent representation of $\mathbf{x}_s$ generated by the encoder of a TST model |
| $\widetilde{\mathbf{x}}_{s_1}^{(rec)}$ | Reconstruction sequence with the source style $s_1$ |
| $\widetilde{\mathbf{x}}_{s_1}^{(trf)}$ | Style-shifted sequence having a desired style $s_2$, and the original source style $s_1$, where $s_1 \neq s_2$. |
| $\mathrm{X}_s$ | Textual dataset having style $s$ and $N$ sequences ($\mathbf{x}_s$,...,$\mathbf{x}_N$) |
| $\boldsymbol{E}$ | Encoder of a TST model |
| $\boldsymbol{E}_s$ | Style-specific Encoder of a TST model |
| $\boldsymbol{D}$ | Decoder of a TST model |
| $\boldsymbol{D}_s$ | Style-specific Decoder of a TST model |
| ***Disc*** | Discriminator block of an adversarial TST model |
| $\boldsymbol{Disc}_s$ | Style-specific classifier for style $s$ used in the discriminator block of an adversarial TST model |
| ***Gen*** | Generator block of a TST model consisting of an encoder-decoder network |

## 1.5 Report outline

The outline of the current report can be described as follows.

**Chapter 1** (the current chapter) of this report introduces the scope of the research and main question of this study.

**Chapter 2** firstly provides the necessary background knowledge about neural networks, then, it reviews the previous work on textual style transfer and focuses on the unsupervised adversarial techniques, as this is the main modelling approach we use in this work.

**Chapter 3** firstly presents the datasets used in our experiments. Then it focuses on the modelling approach implemented in the style transfer systems used in this report. It proceeds by explaining the structure of baseline models while specifications of proposed alternative models are left for later chapters. Finally, it discusses the linguistic dimensions under consideration when evaluating TST models and introduces the related metrics.

**Chapters 4** mainly investigates the separation of the style and content in the latent space of RNN-based TST models in the sentiment domain. The work in this chapter addresses research questions 1 and 5 (section 1.2) and the findings of the reported experiments support research contributions 1 and 5 (section 1.3).

**Chapters 5** extends the experiments of chapter 4 and looks into formality and content versus sentiment and content disentanglement in TST and explores the variations across these two style domains. The work in this chapter addresses research questions 1, 2 and 5 (section 1.2). Moreover, the findings of the reported experiments support the contributions 1, 2 and 5 (section 1.3).

**Chapter 6** further investigates the variations across different style domains by conducting a series of experiments to look into how these variations are encoded. Finally, it studies how these variations can affect the TST task. The work in this chapter addresses research

questions 3, 4 and 5 (section 1.2). Moreover, the findings of the reported experiments support the contributions 3, 4 and 5 (section 1.3).

**Chapter 7** discusses the findings, and limitations of the current work and TST scope, and presents the possible future directions of this research.

# Chapter 2

# Background

This thesis focuses on the scope of TST and looks at a range of TST architectures. The current chapter provides the necessary background knowledge for this research by firstly discussing the related necessary neural network concepts and architectures in section 2.1 and then reviewing the related work in section 2.2.

## 2.1 Background neural network knowledge

The discussions in this section mainly revolve around sequence-to-sequence generation strategies applied by various TST architectures. However, these strategies are not only applicable to TST frameworks, but also to other sequence generation tasks such as neural machine translation, language modelling and image caption generation.

The original neural network architectures are Feed-Forward neural networks (section 2.1.1) that learn non-linear mappings from input to output inspired by neuroscience. The challenge with these networks is that they have fixed width and therefore can only handle fixed length data which led to the development of Recurrent Neural Networks. These Networks, in spite of being capable of handling different length input, have limitations with handling long distance dependencies. Therefore, GRU and LSTM variants of Recurrent

Neural Networks were introduced which performed better in dealing with long sequences as compared to standard Recurrent Neural Networks (section 2.1.2). More recently, transformer architectures have been introduced which apply self-attention strategy to learn the relation ship between the tokens within a given input (section 2.1.4.3). Even though, unlike Recurrent Neural Networks, transformers cannot handle different length data, they generally perform better in capturing semantic and syntactic information from across sentences as a result of using self-attention strategy (Vaswani et al., 2017; Devlin et al., 2018; Kelleher, 2019).

### 2.1.1 Feed-Forward neural network

Feed-Forward neural networks (FFNN) or multilayer perceptrons (MLPs) are considered the basis of some neural networks including Convolutional Neural Networks (CNN) and are applied in many other neural networks such as Recurrent Neural Networks (RNN) or transformers. Taking an input $x$, they aim to approximate a function $y = f_\theta(x)$. For instance, an FFNN-based classifier learns to map an input $x$ to the the correct class $y$ by approximating the parameter $\theta$. The depth and width of an FFNN is the number of the layers and the units in hidden layers respectively (Goodfellow et al., 2016).

FFNNs are acyclic graphs where information flows in a forward direction. The network consists of 3 types of layers: input-layer, hidden layer and output-layer. Figure 2.1 shows an FFNN with an input-layer of size $N$, one hidden layer of size 3 and output layer of size 2. The units of each hidden layer and output layer can be computed by doing the following two steps.

First, computing preactivated values of a given layer by doing the matrix multiplication of the weights corresponding to this layer and the outputs of the previous layer and adding a bias score to them (while computing the first hidden layer, the outputs of the previous layer are the input tokens). For instance, neuron $h_1$ in the hidden layer of the figure 2.1 is

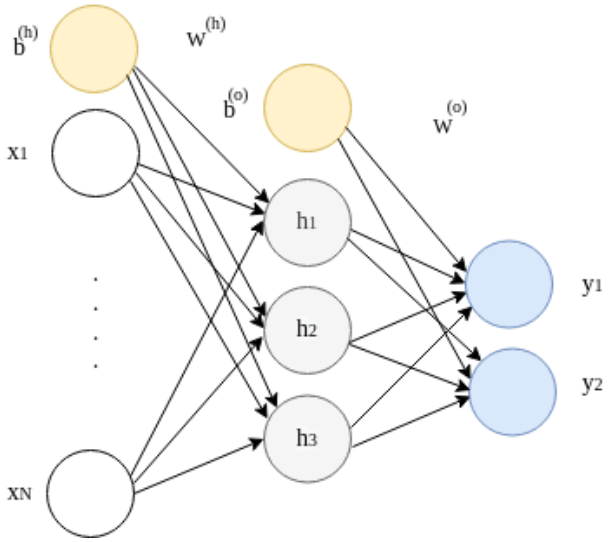


Figure 2.1 The schema of FFNN network

calculated as the following equation shows.

$$preactivated(h_1) = \sum_{i=1}^{N} w_{1i}^{(h)} x_i + b^{(h)} \tag{2.1}$$

The second step involves applying an activation function to the preactivated values. Equation 2.2 shows how $h_1$ of figure 2.1 is computed where $g$ is the activation function.

$$h_1 = g(preactivated(h_1)) \tag{2.2}$$

The output layer is computed similarly by using equations 2.1 and 2.2 where $W^{(o)}$ and $b^{(o)}$ are the weight matrix and bias value corresponding to this layer and $h_1$, $h_2$, and $h_3$ are the outputs of the previous layer.

### 2.1.2 Recurrent neural network

One of the limitations of the FFNNs is that they get fixed-sized vectors as input. This can affect the performance of text generation models, such as language models. Recurrent neural

networks (RNNs) have removed this constraint by recursively processing sequences one element at a time.

RNNs process different inputs at different points in time, but the parameters are shared through the processing of the sequence. Sharing parameters is one of the characteristics of RNNs which enables them to handle long sequences. Due to this characteristics, they have played a dominant role in text processing and text generation (Goodfellow et al., 2016). At each time step, for instance $t$, an RNN cell takes the input $x_t$, and the previous hidden state $h_{t-1}$ and outputs $h_t$ which is the updated hidden state at the time step $t$. This is formulated as $h_t = RNN(x_t, h_{t-1})$.

In practice, using simple RNNs, introduced by Elman (1990), in neural architectures can lead to issues such as exploding and vanishing gradients (Bengio et al., 1994). Different strategies have been proposed to tackle these problems such as the gradient norm clipping technique or using variants of the vanilla RNN.

**RNN variants** We describe Long Short-term Memory Networks (LSTM) and Gated Recurrent Unit (GRU) variants of RNN in this section. In the equations used throughout this section $x_t$ denotes the current input, $h_t$ denotes the current hidden state, $h_{t-1}$ denotes the previous hidden state, $c_t$ denotes the current state of an LSTM memory cell, $c_{t-1}$ denotes the state of the memory cell at the previous time step, a capital $W$ denotes a weight matrix, the $\odot$ symbol denotes an element-wise vector product operation, and a $+$ symbol denotes an element-wise addition between vectors. Also the multiplication between matrices and vectors implicitly includes the addition of bias terms, where the weights and the bias terms are all parameters that are learnt during training.

- LSTM is an RNN variant which was introduced by Hochreiter and Schmidhuber (1996) to address the vanishing gradient issue of simple RNN cells. The main difference between the simple RNN and LSTM is the computation of the hidden state $h_t$ at time step t. Specifically, the RNN cell is replaced with an LSTM unit. An LSTM unit

maintains a hidden state ($h_t$) and a memory cell ($c_t$) and uses three gating units (the forget gate $f_t$, the input gate $i_t$ and the output gate $o_t$) to control the flow of information in the memory cell through time and to calculate the new hidden state at each time step (Kelleher et al., 2020). The forget gate uses the current input $x_t$ and hidden state from the previous time step $h_{t-1}$ to decide what information should be removed from the memory cell. It is implemented as a sigmoid layer (the same width of the memory cell) to generate a masking vector that is fed into an element-wise product with the memory cell to generate a new memory cell state. The fact that the forget gate has a sigmoid activation function means that all the components of the masking vector have values between 0 and 1, and the element-wise product results in memory cell values that have a corresponding sigmoid activation near 0 being pushed to 0 (being forgotten) and memory cell values that have a corresponding sigmoid activation near 1 being maintained. Equation 2.3 lists the calculation of the masking vector $f_t$ used by the forget gate and Equation 2.4 lists the updating of the memory cell state by the forget gate.

$$f_t = sigmoid(x_t.W_{fx} + h_{t-1}.W_{fh}) \tag{2.3}$$

$$c'_t = f_t \odot c_{t-1} \tag{2.4}$$

The input gate decides which values are to be stored in $c_t$ and consists of a *sigmoid* layer and a *tanh* layer. Both of these layers take the current input and the hidden state from the previous time step as input. As in the forget gate, a sigmoid layer is used to generate a masking vector, in this case the masking vector indicates which components in the memory cell should be updated with new information (sigmoid activations near 0 indicate that the corresponding memory cell component should not be updated and activations near 1 indicate that they should). Then a tanh layer decides what values can be added to the memory cell. An element-wise product of the outputs of the tanh layer and the sigmoid layer mean that the update values generated by the tanh layer

are filtered by the sigmoid activations before being added to the memory cell state. Equation 2.5 defines the input gates *sigmoid* layer which defines which values in the memory cell are updated.

$$i_t = sigmoid(x_t.W_{ix} + h_{t-1}.W_{ih}) \tag{2.5}$$

Equation 2.6 defines the input gates *tanh* layer, which defines the values that can be added to cell state.

$$\widehat{c_t} = tanh(x_t.W_{cx} + h^{t-1}.W_{ch}) \tag{2.6}$$

The next step is updating the cell state created by the update from the forget gate $c'_t$ by adding the vector created by the elementwise product of the sigmoid and tanh activations in the input gate, this is shown in equation 2.7.

$$c_t = c'_t + (i_t \odot \widehat{c_t}) \tag{2.7}$$

Finally, the output gate decides what we are going to output, shown in the Equation 2.8.

$$o_t = sigmoid(x_t.W_{ox} + h^{t-1}.W_{oh}) \tag{2.8}$$

The new hidden state $h_t$ is calculated as shown in equation 2.9.

$$h_t = o_t \odot tanh(C_t) \tag{2.9}$$

Importantly, the design of the LSTM unit is such that, during training, the error gradients with respect to the memory cell state are not repeatedly multiplied by a weight that is shared across time steps. So, these gradients are stable and the model is

consequently better able to learn long-distance dependencies through time (Kelleher et al., 2020).

- Gated Recurrent Unit (GRU) Chung et al. (2015) cells are a simplified version of LSTM cells. In contrast to LSTMs which maintains both a hidden state ($h_t$) and a memory cell ($c_t$) and uses three gating units (the forget gate $f_t$, the input gate $i_t$ and the output gate $o_t$), the GRU only maintains a hidden state ($h_t$) and uses only two gates to control the flow of information in hidden state through time and to calculate the new hidden state at each time step. These two gates are known as the reset gate $r$ and the update gate $z$. At each time step $t$ each of these gates use the current input $x_t$ and the previous hidden state $h_{t-1}$ to generate a vector mask the same width as the hidden state: $r_t$ and $z_t$ . These vector masks are then used to transform the previous hidden state $h_{t-1}$ to the current hidden state $h_t$. The integration of the vector masks generated by these two gates with the previous hidden state is designed so that when a component of the reset gate mask $r_t$ is close to 0 the hidden state is forced to ignore the information in the corresponding component of the previous hidden hidden state and to reset that component of the hidden state with the current input (thereby allowing the hidden state to drop information that is no longer relevant), and the update gate controls whether the hidden state is to be updated with a new hidden state $\hat{h}_t$.

  The reset gate vector mask at time $t$, $r_t$, is calculated as shown in equation 2.10.

$$r_t = sigmoid(x_t.W_{rx} + h_{t-1}.W_{rh}) \tag{2.10}$$

  Once the reset gate vector mask is calculated it is then integrated with the previous hidden state $h_{t-1}$ to create a candidate new hidden state $\hat{h}_t$, see equation 2.11. In this equation notice that the element-wise product of the reset gate mask ($r_t$) with the output of the result of the linear layer operation on $h_{t-1}$ results in the $r_t$ mask filtering the

information carried forward from $h_{t-1}$ to $\hat{h}_t$. This carried forward information is then added to the result of the linear layer applied to the current input $x_t$.

$$\hat{h}_t = tanh(x_t.W_x + r_t \odot (h_{t-1}.W_h)) \tag{2.11}$$

The update gate vector mask at time step $t$, $z_t$, is calculated in a similar manner to the calculation of the reset gate mask, see equation 2.12.

$$z_t = sigmoid(x_t.W_{rx} + h_{t-1}.W_{rh}) \tag{2.12}$$

The update gate mask is then used to control both how much of $h_t$ should be retained information from $h_{t-1}$ and how much of $h_t$ should be information from the new candidate hidden state $\hat{h}_t$. Equation 2.13 shows how $z_t$ is used to achieve this. When $z_t$ is near 1 relatively little information from $h_{t-1}$ is retained in $h_t$ and a lot of information from $\hat{h}_t$ is added to $h_t$. Conversely, when $z_t$ is close to zero a lot of information is retained from $h_{t-1}$ and relatively little information from $\hat{h}_t$ is used in the hidden state update.

$$h_t = (1 - z_t).h_{t-1} + z_t.\hat{h}_t \tag{2.13}$$

### 2.1.3 RNN-based language model

Language models are trained to predict the next likely word in a sequence based on the preceding context. RNN-based language models employ vanilla RNN or its gated variants LSTM and GRU as cell units and they include: 1. an embedding layer, 2. hidden layer(s) of RNN, and 3. a projection and a softmax layer which together form the output layer.

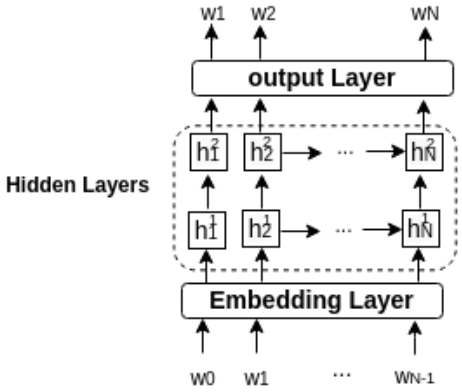


Figure 2.2 The schema of an RNN-based Language Model including 2 hidden layers.

Given an input sequence $x$=[$w_1$,...,$w_N$], as figure 2.2 illustrates an RNN-based language model first projects the tokens $w_i$ to their embedding vector through the embedding layer. Then, it calculates the hidden states of the RNN hidden layers. The output of the k-th hidden layer (k $\in \{1,2\}$ in the model of figure 2.2) at time step $t$ ($t \in \{1,2,...,N\}$) is computed by equation 2.14 where $h_t^k$ is the hidden state of the hidden layer $k$ at time step $t$. In this equation, $h^0$ refers to the embedding layer and the corresponding value for $h_t^0$ is the embedding vector of token $x_t$. $W^k$ and $U^k$ are the learnable weights for the hidden layer $k$. Function $a(.)$ is a non-linear activation function.

$$h_t^k = a(h_{t-1}^k U^k + h_t^{k-1} W^k) \tag{2.14}$$

The output layer, as defined by equation 2.15, predicts tokens at each time step $t$ by applying a linear projection and a softmax layer over outputs of the final hidden layer. In this equation $W^o$ is the weights of the projection layer.

$$Pr(x_t[x_1, x_2, ..., x_{t-1}) = softmax(h_t^k W^o) \tag{2.15}$$

### 2.1.4 Sequence-to-sequence architectures

A sequence-to-sequence network takes a source sequence $x$=[$w_1$,...,$w_N$] as the input and generates an output sequence $y$=[$y_1$,...,$y_M$] (depending on the task, a translated form of $x$ in the desired language in the Neural Machine Translation (NMT), a simplified form of $x$ in a text simplification task, etc). In the text style-shift problem, the target sequence is a rewritten form of $x$ with the desired target style which is supposed to be different from the source style.

As figure 2.3 shows a sequence-to-sequence model relies on an encoder-decoder architecture where the encoder aims at creating a vector representation $z$ of the input and the decoder takes $z$ and generates an output sequence in a manner similar to a language model by conditioning the generation of the target tokens on $z$ and the previously generated tokens. Different neural architectures can be employed for the encoder and decoder, including RNN networks (section 2.1.4.1), convolutional neural network (CNN) (Kaiser et al., 2018) or self-attention networks such as transformer architectures (Vaswani et al., 2017).

Since this type of architecture is at the heart of our work, the remainder of this section focuses on describing various sequence-to-sequence encoder-decoder frameworks.

#### 2.1.4.1 RNN-based sequence-to-sequence architectures

RNN-based sequence-to-sequence architecture proposed by Sutskever et al. (2014) to frame the NMT task can be used in any similar generation task. RNN-based sequence-to-sequence models employ RNNs as encoder and decoder components. The encoder is responsible for encoding the input sequence into a fixed-size vector $z$ and the decoder aims at decoding this vector.

Creating fixed-length representations for variously sized inputs can, however, in practice lead to issues such as losing the input information during the generation process especially in the case of long input sequences. This is mainly due to the fact that the contextual information gets diluted as the encoder processes the tokens along a given input sequence.

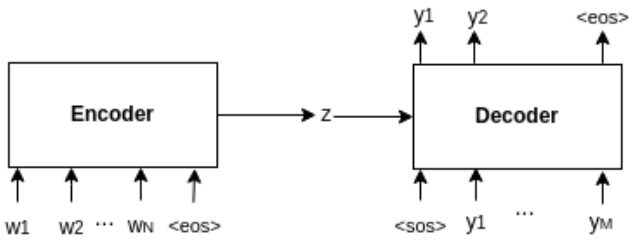


Figure 2.3 The schema of a sequence-to-sequence encoder-decoder network

Sequence-to-sequence architectures which employ attention mechanism (Bahdanau et al., 2015) have managed to overcome this issue to a great extent.

#### 2.1.4.2 Applying input-output attention mechanism in sequence-to-sequence models

The major problem while using RNNs as an encoder network for text processing is that as the time steps proceed the information of earlier steps fades away. Employing different strategies can help the network encode the contextual features better. For instance, reversing the input sequences or employing bidirectional RNNs where one RNN encodes the input information in the forward direction and one reads the input in the backward direction. Using the attention mechanism proposed by Bahdanau et al. (2015), especially in the case of long sequences, has also proved to be an efficient technique to encode the contextual information.

This attention mechanism revolves around the idea of computing the input representation vector dynamically at each generation step instead of providing the decoder with a static representation of the input. We refer to this mechanism as input-output attention throughout this thesis. Applying this mechanism, given an input $x$=$[w_1, ..., w_T]$, its encoded representation (context vector) is calculated as the weighted summation of the encoder states $h$=$[h_1, ..., h_T]$ where the weights are computed dynamically at each generation step. These weights represent the attention. They are computed through a score function which studies the relevance of a given vector $h_j$ with the current state of the decoder $s_{t-1}$ (at time step $t$).

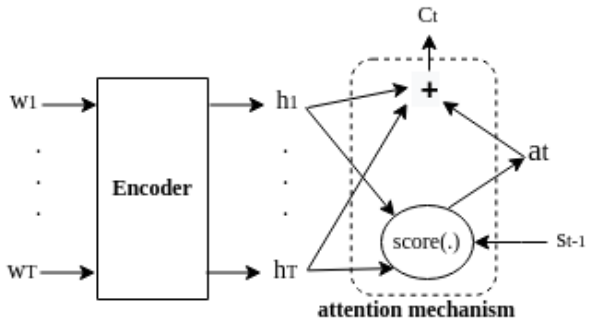


Figure 2.4 Computing $c_t$ at the generation step $t$ by considering the decoder state $s_{t-1}$ and encoder states $h_1, ..., h_T$ corresponding to the input $x = [w_1, ..., w_T]$, $a_t = [a_1, ..., a_T]$ denotes the attention weight vector computed for the encoder states.

The followings are three of the most widely applied methods of how the score function has been defined where $V$, $W_1$ and $W_2$ are trainable weight matrices:

- Dot-Product: $score(h_j, s_{t-1}) = {h_j}^T s_{t-1}$
- Bilinear: $score(h_j, s_{t-1}) = {h_j}^T W_1 s_{t-1}$
- Additive: $score(h_j, s_{t-1}) = V^T tanh(W_1 h_j + W_2 s_{t-1})$

Each weight is then normalized using a softmax layer such that all attention weights sum to 1 leading to the equation 2.16.

$$a_{jt} = \frac{\exp(score_{j,t})}{\sum_{k=1}^{T} \exp(score_{k,t})} \tag{2.16}$$

For the input $x$, equation 2.17 shows how to compute the context vector $c_t$ at each generation step $t$ as the weighted summation of the encoder states (figure 2.4).

$$c_t = \sum_{k=1}^{T} a_{kt} h_k \tag{2.17}$$

While applying input-output attention mechanism, for a generation step $t$, the decoder conditions the generation of the output $y_t$, on $c_t$, the previous hidden state of the decoder $s_{t-1}$ and the token generated at previous time step $y_{t-1}$, i.e. $y_t = f(y_{t-1}, s_{t-1}, c_t)$

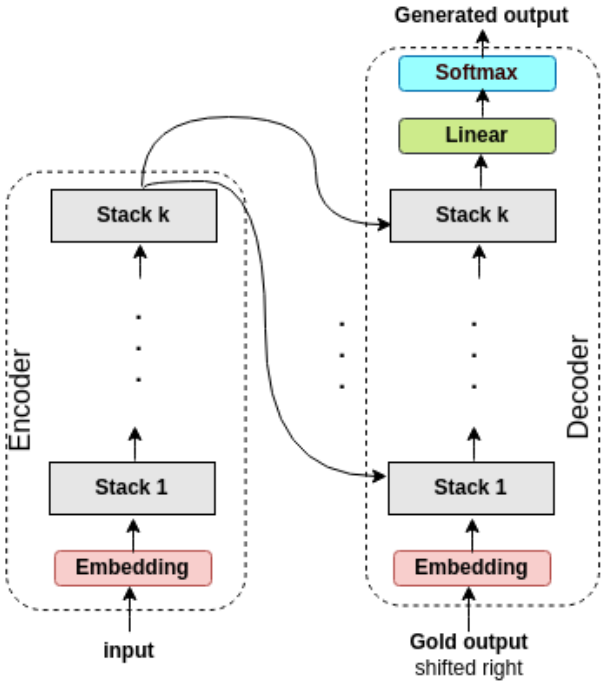


Figure 2.5 The schema of transformer-based sequence-to-sequence encoder-decoder network

#### 2.1.4.3 Transformer-based sequence-to-sequence architecture

Transformer refer to a new type of layer for sequential data. By extension, the term also refers to models that include such a layer. Originally, transformer layers have been proposed in an encoder-decoder sequence-to-sequence architecture where the encoder and decoder both rely on transformers (Vaswani et al., 2017; Lewis et al., 2020; Raffel et al., 2020). Alternatively, transformer models have been proposed to act only as encoders (Devlin et al., 2018; Reimers and Gurevych, 2019) or as decoders (Radford et al., 2018a, 2019; Brown et al., 2020).

The encoder and decoder of a transformer-based encoder-decoder architecture as figure 2.5 shows consist of an embedding layer followed by $k$ stacks of encoding and decoding networks where $k$ is a hyperparameter. The $k$ encoding stacks are identical networks and the same holds for the $k$ decoding subnetworks. Decoder subnetwork also includes a projection layer as the final building block which uses the output of the last stack of the decoder to generate the outputs.

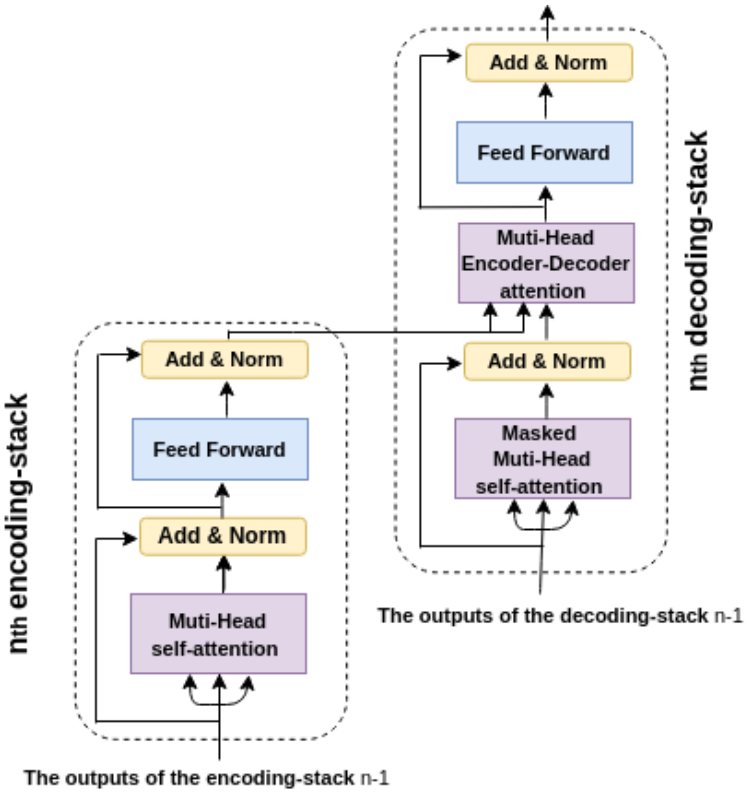


Figure 2.6 The $n_{th}$ encoding-stack and decoding stack of the transformer encoder-decoder (when $n = 1$, the outputs of the embedding layer is fed to the self-attention module). The schema is adapted from the schema proposed by Vaswani et al. (2017).

Each stack of the transformer-encoder as figure 2.6 shows consists of a Multi-Head self-attention module, a Feed-Forward module and two Add and Norm layers which are applied around each of these modules. In addition to these modules, each of the identical stacks of the decoder contains a Multi-Head encoder-decoder attention module and another Add and Norm around this layer. In this section we describe these sublayers as the major building blocks of a transformer network.

**Embedding layer** To create the vector representation of the source input tokens, and target gold tokens, an embedding layer is used in encoder and decoder of the transformer which projects the tokens to their embedding representations similar to any other NLP task. Target gold text exist in case of using parallel data for training. This embedding layer learns both

positional and contextual embeddings for the tokens. This is mainly due to the architecture of transformers which unlike RNN networks cannot encode positional information of the tokens (Vaswani et al., 2017).

Given an input $x = [w_1, ..., w_T]$, the embedding of the token $w_i$ is computed as summation of contextual and positional embeddings. The positional embeddings can be either fixed or learnable (Gehring et al., 2017; Vaswani et al., 2017).

**Fully connected Feed-Forward module** As figure 2.6 shows both encoding and decoding stacks of the transformer encoder-decoder architecture contain a Feed-Forward module (FF) which is applied after their attention-based modules. Given an input $u$, this layer projects it through two dense layers where the first layer uses a *ReLu* activation function. This is formulated as shown in equation 2.18 where $W_u$, $b_u$, $W_o$ and $b_o$ are the learnable parameters of this module.

$$FF(u) = (ReLu(uW_u + b_u))W_o + b_o \tag{2.18}$$

**Residual and normalization module** Residual and normalization module, Add & Norm layers in figure 2.6, are used around each sub-component of the encoding and decoding stacks of the transformer encoder-decoder network, i.e. each encoding and decoding stack includes two and three Add & Norm layers, respectively. This module is applied to help the training process and convergence of the model where using it improves the results.

**Multi-Head attention module** The Multi-Head attention module of the transformer, a self-attention network (SAN), takes the three vectors of query $Q$, value $V$ and key $K$ as input and creates the weighted sum of the values $V$ as the output shown by equation 2.19 where $\alpha$ shows the attention weights corresponding to each of these values and the softmax layer

produces the attention weight distribution for the context generation.

$$Attention(Q,K,V) = softmax(\alpha)V \tag{2.19}$$

$\alpha$ is the correlation between each query and key $\alpha = score(Q,V)$ and as equation 2.20 shows it is computed using the scaled dot-product attention scoring function.

$$score(Q,V) = \frac{QK^{\intercal}}{\sqrt{d_k}} \tag{2.20}$$

In this equation $d_k$ is the dimension of the key vector. The dimension of the query vector is also set to $d_k$ to allow for the dot-product between these two vectors. The division by $\sqrt{d_k}$ is done to stabilize the results by scaling the result(Vaswani et al., 2017).

Building on this single attention mechanism, Vaswani et al. (2017) proposed a Multi-Head attention technique for the transformer architecture. This mechanism involves linear projection of the vectors of $K$, $Q$ and $V$ for $N_h$ times where $N_h$ is the hyperparameter which determines the number of heads. For each of these projected triples of the vectors $K$, $Q$ and $V$, i.e. for $head_i$ ($1 < i < N_h$), the single attention mechanism is applied $head_i$ = $Attention(QW_i^Q, KW_i^K, VW_i^V)$ using the equation 2.19 ($W_i^Q, W_i^K$, and $W_i^V$ are learnable parameters). The Multi-Head attention function is formulated as follows where $W^o$ is a weight matrix.

$$multihead(Q,K,V) = concat(head_1, ..., head_{Nh})W^o \tag{2.21}$$

In each stack, both encoding and decoding subnetworks include a Multi-Head self-attention module (figure 2.6) which computes the attention weights of the tokens within one sequence. To show the formal representation, we consider the n-th stack of the transformer-encoder as well as the output of the previous stack of the transformer-encoder, $e_{si_{n-1}}$, and represent the output of the Multi-Head self-attention module as $MHA$ after applying the Add

& Norm layer to it using the equation 2.22, where in the case of $n = 1$, the outputs of the embedding layer is considered as the inputs of the Multi-Head self-attention module.

$$e_{so_n} = Add\&Norm(e_{si_{n-1}} + MHA(e_{si_{n-1}}, e_{si_{n-1}}, e_{si_{n-1}})) \tag{2.22}$$

The output of the $n_{th}$ encoding stack can be formulated as equation 2.23 where $e_{so_n}$, computed by equation 2.22, is the input of the Feed-Forward module.

$$e_{si_n} = Add\&Norm(e_{so_n} + FFN(e_{so_n})) \tag{2.23}$$

The Multi-Head self-attention module in decoding stacks work similarly to that of the encoding stacks. However, to prevent the attention mechanism from cheating while accessing the gold generated data, a masking strategy is used in the Multi-Head self-attention module of the decoding stacks. Considering the generation step *i*, this strategy sets the positions where $step > i$ to $-inf$ before applying the softmax step in the self-attention calculation.

Transformer architecture also computes the encoder-decoder attention weights, i.e. the attention of the input and generated tokens with regards to each other (Multi-Head encoder-decoder attention module in figure 2.6). This module is used only in the decoding stacks and considering the $n_{th}$ stack, it is computed using equation 2.24 where $d_{so_n}$ is the output of the Multi-Head self-attention module of the decoding stacks after applying the Add & Norm layer to it and $e_{si_n}$ is the output of the $n_{th}$ encoding stack.

$$d_{ed_n} = Add\&Norm(d_{so_n} + MHA(d_{so_n}, e_{si_n}, e_{si_n})) \tag{2.24}$$

The last building block of the transformer-decoder is a projection layer with a softmax activation function (figure 2.5) which is employed to convert the output of the last decoding stack into output probability distributions over the target vocabulary which is the target-style vocabulary in the TST experiments of the current research.

#### 2.1.4.4 Training objectives

Sequence-to-sequence text generation systems can be considered as conditional language models which are trained to model the probability $Pr(y|x)$, i.e. the probability of creating the generated sequence $y$ based on the given input sequence $x$. In the case of supervised training, the training objective of the model is to minimize the negative log-likelihood of equation 2.25 (across the training corpus).

$$\mathscr{L}(\theta_E, \theta_D) \quad = \quad -\log \Pr(\mathbf{y}|\mathbf{x}) \tag{2.25}$$

This equation shows the cross-entropy between the input sequence $x$ and the output sequence $y$ where$\theta_E$ and $\theta_D$ are the trainable parameters of the encoder and decoder subnetworks which are estimated during the training.

In the case of training the sequence-to-sequence models in an unsupervised manner, the training objective involves optimizing some additional losses. For instance, while applying approaches similar to GANs (Goodfellow et al., 2014) which broadly speaking include a classifier to guide the training process, the training objective consists of an additional adversarial loss

#### 2.1.4.5 Inference algorithms

While doing a sequence-to-sequence text generation task, the model is trained to maximize the probability $Pr(y|x)$, i.e. the probability of the generated text $y$ given the input sequence $x$. During generation, the decoder computes the probability distribution of each generated token conditioned on the previous outputs which at time step $t$ it is denoted as $Pr(\widetilde{w}_t|\widetilde{w}_1, ..., \widetilde{w}_{t-1}, z)$ where $z$ is the input latent space. The probability distribution of the generated token $\widetilde{w}_t$ over the vocabulary $V$ is computed by equation 2.26 which applies a projection layer to the hidden state of the decoder and then applies a softmax layer to the result.

$$\Pr(\widetilde{w}_t|\widetilde{w}_1,...,\widetilde{w}_{t-1},z) \quad = \quad softmax(W_o s_t + b_o) \tag{2.26}$$

At each generation step, there are $|V|$ (size of the vocabulary) options as the generated token. Considering all these tokens at each generation step becomes computationally expensive. Therefore, different search algorithms can be applied while doing the generation steps.

**Search algorithms** Greedy search and Beam search have been widely applied as search algorithms during generation. Greedy search which is a special case of the beam search considers only the tokens with the highest probabilities from the probability distribution of the vocabulary list at each step of decoding (equation 2.27).

$$w_t = \quad argmax_{w \in V} \Pr(w|\widetilde{w}_1,...,\widetilde{w}_{t-1},z) \tag{2.27}$$

Beam search algorithm keeps track of $k$ tokens with the highest probability at each time step where $k$, the beam size, is a parameter. This leads to a lattice of tokens based on which the best path is returned. Beam search functions the same as greedy search in case $k$ is set to 1.

## 2.2 TST related work

Developing a better understanding of the concept of style is necessary while dealing with the TST task. One of the reasons of this significance is that how the concept of style is viewed can inform the TST modelling approaches. A group of approaches consider style as an independent element from the content which can be defined explicitly, whereas the second approach considers style as a holistic concept and an integral component of a text (Tikhonov

and Yamshchikov, 2018). Based on this distinction, we categorize and review the previous research in the sections of 2.2.1 and 2.2.2.

### 2.2.1 Approaches informed by an explicit concept of style

TST approaches which are informed by an explicit concept of style follow two general steps: first, removing the style from the input text, then, generating the style-shifted output text.

**Filtering out style markers of inputs** As a preprocessing step, the style markers of the input sequences are detected and filtered out. To detect the style markers, different computational approaches and frequency-based techniques, such as TF-IDF, have been applied (Li et al., 2018a; Madaan et al., 2020). Alternatively, some previous work have introduced style marker detectors by employing neural network classifiers (Leeftink and Spanakis, 2019; Xu et al., 2018; Zhang et al., 2018a; Sudhakar et al., 2019) and attention-based techniques (Bahdanau et al., 2015) where these style marker detectors adopt different neural network architectures such as LSTM or transformers (Xu et al., 2018; Zhang et al., 2018a; Sudhakar et al., 2019).

**Generating style-shifted sequences** To generate style-shifted sequences, previous work has employed retrieval strategies, neural approaches or a combination of these two techniques. Retrieval approaches directly extract the corresponding style-shifted text for a given input from a corpus of the target style as the text which resembles the most to the style-free representation of the input (Li et al., 2018a). Alternatively, these approaches create a style-shifted text by first retrieving the style markers from a corpus of the target style and then directly concatenating them with the style-free representations of the given input. However, style-shifted texts created by following the the latter strategy do not have very high fluency (Ramos, 1999; Li et al., 2018a; Leeftink and Spanakis, 2019). To address this issue, neural encoder-decoder architectures can be employed together with retrieval strategies to

generate style-shifted outputs. Doing this, the generation of the style-shifted text for a given input, is conditioned on the style-free representations of the input and the retrieved target style segments (Li et al., 2018a; Sudhakar et al., 2019; Zhang et al., 2018a; Xu et al., 2018). The generation process can also be guided by conditioning the generation on the style-free input representations and target style (Sudhakar et al., 2019).

While taking retrieval approaches, to extract sequences and segments from a corpus of a desired style, vector representations of the text are created employing TF-IDF-based or embedding-based techniques. Embedding-based techniques rely on using pre-trained embedding models which create sequence embedding including Universal Sentence Encoder (Cer et al., 2018) or pre-trained models which create token embeddings such as GloVe (Pennington et al., 2014) where a pooling technique is needed to create the sequence vectors (Ramos, 1999; Li et al., 2018a; Leeftink and Spanakis, 2019; Sudhakar et al., 2019).

### 2.2.2 Approaches informed by an implicit concept of style

Some previous research considers style as an implicit concept where it can be defined as the consistent variation between the texts under study. This view makes the definition reliant on the discrimination, i.e. considering two corpora, style is the information which consistently differentiates them, while being invariant within each corpus. In other words, in this approach style is fundamentally connected to the concept of content and each style can be taken as a separate language (Tikhonov and Yamshchikov, 2018; Jin et al., 2022).

Taking this view, previous research used supervised techniques to train the TST systems (section 2.2.2.1) or in the case of the absence of the parallel data or limited access to it, they have used unsupervised (section 2.2.2.3) or semi-supervised strategies (section 2.2.2.2).

#### 2.2.2.1 Training using parallel data

In recent TST research a popular approach is to adopt end-to-end learning techniques to deal with this task, following strategies similar to a supervised NMT task. Models are end-to-end encoder-decoders which directly translate a text $x_{s1}$ having the style $s_1$ (source language) to a text $x_{s2}$ having the style $s_2$ (target language) where the output text should be grammatically fluent and resemble the inputs in terms of content.

These NMT like TST systems have been developed to transfer style using parallel data from various domains, for instance, simplification (Ma and Sun, 2017), summarization (Ma and Sun, 2017; Rush et al., 2015) and formality (Xu et al., 2019; Rao and Tetreault, 2018). These sequence-to-sequence encoder-decoder models have been mostly based on either RNNs (Sutskever et al., 2014; Bahdanau et al., 2015) or transformers architecture (Vaswani et al., 2017). For example, Ma and Sun (2017) apply a LSTM-based encoder-decoder architecture, Rush et al. (2015) use an attention-based RNN network, and Xu et al. (2019); Rao and Tetreault (2018) use transformer-based models to deal with the formality-shift task where they focused on improving the content preservation power of the TST systems using additional losses.

Moreover, different techniques have been proposed to create pseudo-parallel data to get around the issue of the deficit of labelled data. This thesis proceeds by describing some of these methods in more detail (section 2.2.2.2).

#### 2.2.2.2 Addressing the deficit of parallel data

Limited access to parallel data is a major issue in the TST field and different techniques has been previously proposed to alleviate this problem. For instance, Johnson et al. (2017) applied zero-shot translation method which uses intermediate resources to facilitate parallel training, i.e. it uses different languages as the pivot language in NMT task to enable the translation in between the two languages for which little or even no parallel data is available.

This strategy can be similarly applied for TST by using different styled data as intermediate resources (Carlson et al., 2017).

Moreover, different techniques have been proposed to create pseudo-parallel data to get around the issue of the deficit of labelled data. Here, we briefly describe some approaches that can be used to create TST pseudo-parallel data which can be applied to train models in parallel mode. Some of these methods use back translation technique to construct pseudo-parallel data in different *NLG* problems such as NMT (Sennrich et al., 2016; Prabhumoye et al., 2018) or TST (Zhang et al., 2020). The idea is to use monolingual text and create its pseudo parallel counterpart. To clarify more, Zhang et al. (2020) created pseudo-parallel formal text by following these steps. For a given informal data $x_{inf}$, they did a cycle of translation to a pivot language and back to English using a NMT system trained on the formal data. The resulted text $x_f$ is the pseudo parallel text of $x_{inf}$ having the same content but formal style.

Interestingly, TST can also help other tasks, for instance, some previous research implemented TST techniques to augment the parallel data, i.e. given a monolingual data, $X_{s1}$, with a style $s_1$, they use a TST system trained in this style domain and generate pseudo-parallel data $X_{s2}$ with style $s_2$ (Zhang et al., 2018b).

Retrieval-based strategies can also be applied to augment the parallel data where the idea is that the text retrieved from the two different-styled corpora can be parallel if they are semantically very similar. Therefore, in this approach, given a text, $x_{s1}$, with a style $s_1$, different techniques are applied to compute its semantic similarity to the sequences of a monolingual corpus $X_{s2}$ with style $s_2$. Sequence $x_{s2}$ having the highest similarity with $x_{s1}$ is labelled as its pseudo-parallel counterpart (Jin et al., 2019).

In spite of these approaches, composing enough data to enable the parallel training is still challenging. This is why it has been proposed to remove the need for parallel data, through so-called unsupervised approaches. Section 2.2.2.3 mainly focuses on these approaches.

#### 2.2.2.3 Unsupervised training using adversarial techniques

In the absence of the parallel data, various unsupervised techniques have been employed for TST. They have mostly focused on adopting end-to-end learning strategies and proposed frameworks which contain a generator block and a style-shifting block.

**Generator block** The generator block is similar to TST systems trained by supervised techniques (section 2.2.2.1). It consists of a sequence-to-sequence encoder-decoder network which creates latent representations of inputs and generates output texts meeting the requirements of the task. In TST research, the encoder and decoder subnetworks are frequently implemented as either RNN-based architectures (Sutskever et al., 2014; Bahdanau et al., 2015), such as the TST systems proposed by Shen et al. (2017); Singh and Palod (2018); Fu et al. (2018a); Romanov et al. (2019) or different variants of the standard RNN-based architectures. Some systems use style-specific decoder TST frameworks where multiple decoder subnetworks share an encoder (Fu et al., 2018a), other systems use variational encoder TST models which condition the generation of the output on a vector sampled from the posterior distribution of the latent space whose parameters are predicted by the encoder (Hu et al., 2017a; John et al., 2019) and yet other systems use style-specific encoders which include one encoder for each different-styled monolingual corpus (Jafaritazehjani et al., 2021). The generator block can also be based on the variants of transformer network (Vaswani et al., 2017) such as the model proposed by (Dai et al., 2019b).

The encoder subnework of the generator block is responsible for creating the vector representation of the input text and the decoder subnetwork generates output text, typically conditioned on the input latent representation concatenated with an embedding of the target output style. For a given input, if the input style and target style are the same, the generator block becomes an auto-encoder and the decoder subnework reconstructs the input, otherwise, the decoder creates a style-shifted paraphrase of the input text.

**Style-shifting module** To reach this goal different strategies have been previously proposed.

Adversarial techniques have been widely implemented in the recent years to enable training the TST systems in an unsupervised manner (Ma and Sun, 2017; Shen et al., 2017; Singh and Palod, 2018; Fu et al., 2018a; Romanov et al., 2019; Hu et al., 2017a; John et al., 2019). Similar to Generative Adversarial Networks (GAN) (Goodfellow et al., 2014), adversarial TST models include a discriminator and a generator block where the discriminator plays the role of the style-shifting block. The discriminator block contains classifiers specific to each style and these classifiers can either be trained together in parallel with the generator block (Shen et al., 2017; Dai et al., 2019b; Romanov et al., 2019; John et al., 2019; Fu et al., 2018a; Tikhonov et al., 2020; Li et al., 2020; Zhao et al., 2018) or can be pre-trained networks (Prabhumoye et al., 2018; Hu et al., 2017b; Yamshchikov et al., 2019; Romanov et al., 2019; John et al., 2019).

The classifier(s) can be fed by the output generated by the decoder(s) of the TST system where the goal of discriminator block is to distinguish between the style-shifted text and reconstructed text or human-generated text. During training, if the discriminator block detects that an input is style-shifted, the generation process is penalized to push the generator block to construct outputs that appear more similar to the human-created text or reconstructed sequences (Shen et al., 2017; Dai et al., 2019b; Prabhumoye et al., 2018). In some systems a discriminator block can also be applied to the latent representations of the input created by the encoder subnetwork. Typically, this is done to encourage the encoder to create style free latent representations of the input and this is achieved by penalizing the system if the discriminator block can correctly label a latent representation with the source style of the input (Romanov et al., 2019; John et al., 2019; Fu et al., 2018a; Tikhonov et al., 2020; Li et al., 2020; Zhao et al., 2018).

Style-shifting block can rely on back-translation technique for removing or loosening the stylistic features of inputs by doing a cycle of translations (Prabhumoye et al., 2018;

Rabinovich et al., 2017; Zhang et al., 2020). Employing this strategy includes using a translation network that for a given input $x$ in language $L_1$, it firstly creates an output in language $L_2$ (where $L_1$ and $L_2$ are different languages). Then, it translates this output back to $L_1$ as $\tilde{x}$ where the stylistic features in $\tilde{x}$ has faded away compared to $x$.

As discussed in this section, adversarial classifiers can be applied in TST systems to guide the generation towards creating text having a target style. However, the application of these classifiers can be extended to ensure the encoding of the content-related information of inputs in their corresponding latent vectors created by the encoder. This leads to higher fidelity of the style-shifted outputs to the content of their corresponding inputs, i.e. applying this strategy can improve the content preservation power of the frameworks (Romanov et al., 2019; John et al., 2019).

### 2.2.3 Studying different aspects of the TST task

Applying end-to-end approaches while framing the TST task enables learning of latent representations of inputs (Kelleher, 2019). Unsupervised TST models have focused on separating style and content in their latent space based on the assumption that style-content disentanglement is possible (Xu et al., 2018; Jin et al., 2022). However, little previous work has studied the latent space of these models. In this research we would like to analyse the latent space of TST systems and extend this analysis across the domains of style.

*NLP* researches have studied various architectures such as transformers to explore how linguistic information is encoded in different layers of these networks. This has led to adopting more informed strategies and resulted in improved performance on *NLP* tasks (Nedumpozhimana and Kelleher, 2021; Nedumpozhimana et al., 2022). However, to the best of our knowledge there has been little research focused on how style is encoded in latent space of different neural TST systems. More importantly, whether variations in the concept of textual style exist across domains and if so how are they encoded within the

latent representations of neural TST systems. In the current work, we investigate these open questions in the TST field.

## 2.3 Summary

The current chapter firstly provided an overview of the basics of the neural networks which are related to the architectures and experiments in the later chapters. Then, it reviewed previous TST work where many researchers have assumed style and content as separable elements of the text. The vast majority of this previous work has mostly employed adversarial end-to-end encoder-decoder architectures as the generator block and considered the latent vectors created by the encoders as style-free representations of the input sequences. However, to the best of our knowledge, this assumption has not previously been investigated. The current research explores the style-content separation across the style domains following the analysis of the extent to which style is encoded within the latent representations generated by the encoders of the various RNN-based and transformer-based TST systems. The thesis then reports on a series of experiments that explore the characteristics of style across a number of style domains. The results of these later experiments point to the fact that the encoding of style can vary across domains. Overall, the findings of this work contribute novel knowledge to the field of TST in terms of foregrounding the importance of examining and understanding the characteristics of style within a domain when designing a TST system and also the selection of appropriate performance metrics for TST in a given domain.

The next chapter of this manuscript describes the data, and proceeds by introducing the state-of-the-art RNN-based and transformer-based TST frameworks used as the baseline systems in the experiments of the chapters 4, 5 and 6. It then describes the experimental and evaluation methodology.

# Chapter 3

# Methodology: data, modelling approach and baseline models, evaluation

This chapter firstly describes the data used in the experiments reported in this thesis (section 3.1). Section 3.2 then provides an overview of the modelling approach which is taken throughout the thesis and describes the models which are applied as the baseline systems in this report. The details of the parameters of these baseline frameworks are provided in section 3.2.4. Section 3.3 focuses on the evaluation methodology (both automatic and manual) and explains the evaluation aspects as well as the evaluation metrics which are applied to investigate the performance of the TST models .

## 3.1 Data

Throughout this research, we study the style domains of sentiment, formality and simplicity. Sentiment which is the binary opinion polarity is studied using the Yelp Restaurant Reviews corpus (section 3.1.1) where positive and negative restaurant reviews form the data. The style domain of formality is studied using the GYAFC corpus (section 3.1.2) where formal and informal text form the dataset. Finally, the Newsla corpus (section 3.1.3) is used to

| | Style | |
|---|---|---|
| | **Formal** | **Informal** |
| Parallel text | dear Sam, a brief note to thank you for your help. | hi Sam, thanks! |
| Non-parallel text | dear Sam, a brief note to thank you for your help. | I've had a blast! |

Table 3.1 Samples of parallel and non-parallel text from the formality domain.

study the style domain of simplicity where simple and complex text form the data. These corpora are used to perform various experiments throughout this research. The sentiment and formality corpora are used to train TST models as well as studying the latent space of various frameworks. The simplification corpus, however, has been mostly applied to conduct experiments to deepen the result analysis.

In the current study, TST is framed in an unsupervised manner using non-parallel corpora which have binary style. This means that each corpus that is used must contain two datasets where sequences of the first dataset have style $s_1$ and sequences of the second dataset have style $s_2$ ($s_1 \neq s_2$). Moreover, the binary styled data of each corpus do not need to be parallel[1] due to applying unsupervised training techniques. In the scope of TST, parallel sequences, as row 1 of table 3.1 shows, differ only in the aspect of style, whereas non-parallel sequences (row 2) differ in terms of both style and content. It is noteworthy that even if the data we use is parallel, we use it in non-parallel mode during the training, i.e. we implement unsupervised strategies for training and treat the parallel texts as if they are non-parallel.

Tables 3.2, 3.3 and 3.5 show the data distributions of the datasets Yelp, GYAFC and Newsla that we use in the later chapters, i.e. they include the size of training, development and test data. They also report the average, maximum and minimum length of the sequences in each binary styled dataset of a corpus. The vocabulary size of each corpus is reported after replacing words occurring less than 5 times with the `<unk>` token considering the training data of that corpus from the both styles $s_1$ and $s_2$.

[1]Terms parallel and non-parallel have been referred to as aligned and non-aligned in some previous research.

### 3.1.1 Yelp restaurant reviews

Yelp Restaurant Reviews is a large-scale corpus consisting of 4.7 million user reviews and is released by Yelp![2], a network where users review businesses such as restaurants, bars, etc. The Yelp dataset has been used for personal, educational, and academic purposes in NLP. The reviews of the Yelp corpus are rated using a five star ranking system and are labelled with regards to these ratings as positive and negative if their corresponding stars are above or below three respectively and three-starred reviews are discarded. This negative and positive texts in this corpus are not parallel. In our experiments, we use two released versions of Yelp dataset which we refer to them as *Yelp-small*[3] (Li et al., 2018a) and *Yelp-large* (Shen et al., 2017)[4]. For *Yelp-small*, Li et al. (2018a) created human-generated style-shifted sequences for the test data. We use this gold parallel data to perform some unigram-based analysis experiments in section 6.3.2.

In both *Yelp-small* and *Yelp-large*, sentence level is taken as the level of data analysis and each sentence is labelled with the label from its corresponding review, i.e. a positive review including five sentences makes five data entries in the positive dataset. This can lead to neutral sentences being labelled as positive or negative especially in the case of long sequences and long reviews. To get around this issue, Shen et al. (2017) filtered out reviews exceeding 10 sentences as well as sequences exceeding 15 tokens. This preprocessing step which is also applied in data we use is based on the assumption that longer reviews are more likely to contain neutral sentences and longer sentences are more likely to be neutral.

Both of these datasets are normalized, taking preprocessing steps such as lower casing, replacing numbers with a special token `<num>` and inserting space between tokens and punctuation as well as between punctuation and punctuation.

[2]https://www.yelp.com/dataset
[3]Distributed under "CC BY-SA 4.0 license".
[4]Distributed under "Apache-2.0 license".

| Data | *Yelp-small* | | *Yelp-large* | |
|---|---|---|---|---|
| Style | Positive | Negative | Positive | Negative |
| Train | 266041 | 177218 | 267314 | 176787 |
| Dev | 2000 | 2000 | 38205 | 25278 |
| Test | 500 | 500 | 76392 | 50278 |
| Avg-len | 8.43 | 9.55 | 8.45 | 9.66 |
| Max-len | 15 | | 15 | |
| Min-len | 1 | | 1 | |
| Vocab-size | 9352 | | 9500 | |

Table 3.2 The data distribution of *Yelp-small* and *Yelp-large*.

### 3.1.2 GYAFC dataset

Grammarly's Yahoo Answers Formality Corpus (GYAFC) (Rao and Tetreault, 2018) contains human-labelled informal and formal sentences which are crawled from two domains: Entertainment & Music (E&M) and Family & Relationships (F&R) in Yahoo Answers[5]. GYAFC[6] is a parallel corpus, i.e. parallel sequence pairs form the formal and informal train, test and development splits of this corpus where each text pair differ only in the style formality (similar to row 1 of the table 3.1). Each sentence of the test set and development set of the GYAFC dataset has four human-generated paraphrases. For instance, for a given text $x$ of style $s_1$, there are four gold paraphrases generated manually by human experts as the gold parallel sequences of $x$ of style $s_2$.

We combine E&M and F&R and label the resulting dataset as *GYAFC-*$v_1$. For our experiments, we, then, modify *GYAFC-*$v_1$ and create a corpus which is referred to as *GYAFC-*$v_2$ throughout this manuscript. The statistics of these datasets are summarized in table 3.3. We employ *GYAFC-*$v_2$ in our experiments in the later chapters as non-parallel corpora by considering the style of each set as the only label available. We only use the gold parallel

[5]https://answers.yahoo.com

[6]This corpus can be accessed upon request for academic research from https://github.com/raosudha89/GYAFC-corpus.

| Data | *GYAFC-$v_1$* | | *GYAFC-$v_2$* | |
|---|---|---|---|---|
| Style | Formal | Informal | Formal | Informal |
| Train | 104562 | 104562 | 102502 | 104044 |
| Dev | 5144 | 5124 | 5064 | 5111 |
| Test | 2101 | 2748 | 2076 | 2739 |
| Avg-len | 11 | 10.3 | 12.4 | 12 |
| Max-len | 41 | 19 | 24 | |
| Min-len | 3 | 6 | 3 | |
| Vocab-size | - | | 11409 | |

Table 3.3 The data distribution of *GYAFC-$v_1$* and *GYAFC-$v_2$*.

text of the test set to perform some unigram-based analysis experiments in section 6.3.2. We describe *GYAFC-$v_2$* in more details in the following paragraphs.

***GYAFC-$v_2$:*** To compose this dataset, we first took some preprocessing steps on *GYAFC-$v_1$* to make the text more consistent by for instance replacing different forms of the same token with one form (steps 1, 3 and 4) and replacing similar tokens and phrases with one special token (steps 2 and 5). The preprocessing steps are listed as follows.

1. Lower casing the tokens of the sequences.

2. Replacing the numbers, website addresses, email addresses and emojis with special tokens: `<num>`, `<website-tok>`, `<email-tok>`, and `<emoji-tok>`.

3. Inserting space between tokens-punctuation and punctuation-punctuation.

4. Making informal data more consistent. Compared to formal data, informal text does not strictly follow language rules. This can lead to presence of different variants for one token. For instance, in table 3.4, there are non-standard forms of *hott* and *hooooooooot* for the token *hot*. To reduce the size of the vocabulary and also the number of `<unk>` tokens, all non-standard forms of highly frequent tokens are converted to one form. In the samples 3 and 4 of table 3.4, for instance, *hott* and *hooooooooot* are converted into

| |
|---|
| 1. ohhh noo! I nooo, that's what I'm saying ..... |
| 2. i no bt i cnt rememba |
| 3. i don't think it's so imporant,'coz she's so hooooooot |
| 4. me ohohohohohoh boy woow hott omg heck yes!!!!!! |

Table 3.4 Informal sample sequences.

*hoott*. Therefore, there is a standard and a non-standard form available for this token in informal data.

To select the non-standard forms, the informal training data was tokenized and the tokens which had a frequency lower than the threshold 5 were checked manually to distinguish between the low-occurring tokens and tokens which were written in a non-standard form.

5. Replacing the sequence of the long sequences of the punctuation into shorter ones; for instance, converting ..... to ... and !!!!!! to !!! (samples 1 and 4 of table 3.4). To detect these tokens, 500 sequences were randomly selected from the informal data and were reviewed manually and tokens with different writing formats were listed. This is to avoid having very long sequences which can be removed while doing the length normalization.

6. Detecting and filtering non-English sequences: To do so, non-English sequences were first detected using python language detector library. Then, these non-English texts were double checked manually to save English sequences which were falsely labelled as non-English. Doing this manual step was important due to the presence of `<unk>` tokens, mainly in informal text, which raises the probability of English texts being labelled as non-English.

7. Removing length outliers: To do so, the box plot of the length distribution of formal and informal data were considered separately, and sequences whose length are beyond the whiskers of these plots are labelled as outliers and removed from data.

Mathematically speaking, outliers were detected following these steps: First, the dataset is divided into half considering the median of the data. Second, the lower quartile $Q_1$ and upper quartile $Q_4$ are computed as the median of the lower half and upper half of the data. Then, the interquartile range is calculated as $IQR = Q_3 - Q_1$. Finally, outliers are the data points out of the range of $[Q_1 - (1.5 * IQR), Q_3 + (1.5 * IQR)]$.

Around 2% of the data of the *GYAFC-v*$_1$ was removed by doing these preprocessing steps. The preprocessed data was then shuffled in order to have a mixed order of the sequences from the two domains of Entertainment & Music (E&M) and Family & Relationships (F&R) in the resulting corpus *GYAFC-v*$_2$.

### 3.1.3 Newsela simplification dataset

*Newsela*[7] corpus contains the data of 1130 news articles. Each article contains 5 versions: 1 original news text labelled as $L_0$ throughout this manuscript and 4 human-generated simplified versions labelled as $L_1$, $L_2$, $L_3$ and $L_4$. They are produced by *Newsela*, a company that creates reading materials for classroom use of pre-college students. This dataset includes parallel textual data where $L_1$, $L_2$, $L_3$ and $L_4$ are simpler versions of a given text $L_0$ which is considered to have the *complex* (*non-simple*) style (table 3.7, samples 1 and 2). Human-generated paraphrases are designed to be readable by children from different age groups. Therefore, they have different levels of simplicity where given an $L_0$ text, $L_1$ is its least and $L_4$ is its most simplified paraphrase. It is worth noting that not all $L_0$ sequences have 4 paraphrases, but all have at least 1 simplified paraphrase (Xu et al., 2015).

We create a simplification corpus where the binary styles are complex and simple using the *Newsela* dataset and refer to it as *Newsela-v*$_1$ throughout this manuscript. The statistics of this corpus is summarized in table 3.5. We describe this corpus in more details in the following section.

[7]This corpus can be accessed for academic research upon request https://newsela.com

| Data | *Newsla-v*$_1$ | |
|---|---|---|
| Style | Complex | Simple |
| Train | 26192 | 24440 |
| Dev | 1000 | 1000 |
| Test | 1000 | 1000 |
| Avg-len | 26 | 14 |
| Max-len | 56 | 25 |
| Min-len | 3 | 3 |
| Vocab-size | 12087 | |

Table 3.5 The data distribution of *Newsla-v*$_1$ dataset.

**Newsela-$v_1$** We created a simplification corpus out of the *Newsela* dataset by using $L_0$ texts as the complex data and $L_3$ and $L_4$ texts as the simplified data. To do this, we first created two complex sets and one simple set using the following five steps and then did the train, test and development splits.

1. If at least one of the $L_3$ or $L_4$ simplified paraphrases are available for a given original $L_0$ text, $L_0$ text will be added to the first complex set. Then, its corresponding $L_4$ paraphrase will be added to the simple set and in case $L_4$ text is not available, its $L_3$ paraphrase will be considered as the simplified paraphrase of the $L_0$ text in the simple set.

   The reason for composing the *simple* data out of $L_4$ texts (or $L_3$ if $L_4$ paraphrase is not available) is to maximize the distinction between the texts of the 2 styles of *complex* and *simple*. As table 3.6 illustrates $L_0$ texts have a high word overlap with $L_1$ texts, 0.6212, and $L_2$, 0.4909 as compared to the $L_3$ and $L_4$ texts where the word overlap drops to 0.3807 and 0.3028. The samples shown in table 3.7 show how sequences $L_1$ and $L_2$ paraphrases can be similar to the $L_0$ texts in terms of style, i.e. they are slightly simpler (if any) compared to the $L_0$ sequences.

2. Otherwise (if neither $L_3$ nor $L_4$ texts are available), the $L_0$ text is included in the second complex set.

| Files | $L_1$ | $L_2$ | $L_3$ | $L_4$ |
|---|---|---|---|---|
| $L_0$ | 0.6212 | 0.4909 | 0.3807 | 0.3028 |

Table 3.6 The word overlap between the file $L_0$ of the Newsla dataset and files $L_1$, $L_2$, $L_3$, and $L_4$

3. The first complex set and the simple set form a parallel corpus from which we randomly select text pairs to form the parallel test sets. The reason for creating a parallel set as the test split is that in section 6.3.2, we need this parallel data to do a unigram-based analysis experiment in the domain of simplicity.

4. The second complex set is then merged with the remainder of the first complex set and gets shuffled. The resulting complex set together with the remainder of the simple set form the non-parallel train and development sets where splitting the data is done randomly.

5. The following preprocessing steps are then taken on the train, test and development splits to form the final corpus[8]:

- Lower casing the tokens of the sequences.
- Replacing the numbers, website addresses, and email addresses with special tokens: `<num>`, `<websitead>`, `<emailad>`.
- Inserting space between tokens-punctuation and punctuation-punctuation.
- Making informal data more consistent, by transforming tokens which have more than one written forms to one form, such as converting ca n't to can't.
- Removing length outliers. Considering the box plot of length of the simple and complex data separately, sequences whose length are beyond the whiskers of these plots are labelled as outliers and removed from data.

[8]Around 1.41% of the complex data and 3.26% of the simple data were removed during the preprocessing steps and the length analysis.

| **From complex to simple** |
|---|
| **Sample** 1**:** |
| $L_0$**. Servicewomen complained that the ban prevents them from advancing their careers .** |
| $L_1$. Servicewomen complained that the ban was limiting their career opportunities . |
| $L_2$. Servicewomen complained that the ban hurts their careers . |
| $L_3$ Servicewomen say that the ban hurts their careers . |
| $L_4$. Military women say that the ban hurt their careers . |
| **Sample** 2**:** |
| $L_0$**. Soldiers who drive fuel trucks , provide medical support or even sort mail can come under fire in modern warfare .** |
| $L_1$. Soldiers who drive fuel trucks , provide medical support or even sort mail can come under fire in this kind of modern warfare . |
| $L_2$. In this kind of modern warfare , soldiers doing any kind of job can come under fire . |
| $L_3$. In this kind of warfare , soldiers doing any job must be ready to fight . |
| $L_4$. Soldiers doing any job must be ready to fight at all times . |

Table 3.7 Examples of the sequences in *Newsela Simplification Dataset* where $L_0$ is the original sequence (*complex* style) and $L_1$, $L_2$, $L_3$ and $L_4$ are the simplified paraphrases. $L_1$ is the least and $L4$ is the most simple versions of $L_0$

## 3.2 Modelling approach

Although each of the experiments we report in the later chapters test multiple architectures, all of these architectures implement a similar unsupervised modelling approach. Consequently, in this section we provide an overview of this modelling approach. To frame the unsupervised textual style transfer, we use adversarial training by following the idea of GANs and employing classifiers as discriminators in our systems which enables the training by just relying on unaligned corpora (differently styled corpora). The main idea behind using this approach is that the discriminators guide the training in the direction of generating the style-shifted sequences in a desired style $s$, so that these sequences cannot be distinguished from the human-generated sequences in the corpus with the style $s$.

We introduce RNN-based and transformer-based adversarial TST baseline models in the following sections. These models are used throughout the experiments of the next chapters of this manuscript.

### 3.2.1 RNN-based TST baseline model

We use the adversarial TST framework proposed by Shen et al. (2017) as our baseline model[9]. This architecture is composed of a generator block and a discriminator block which are described in this section. This section proceeds by explaining the adversarial training regime of this framework.

#### 3.2.1.1 Generator block

The generator block ***Gen*** is based on a sequence-to-sequence model which consists of $(i)$ an encoder **E** and $(ii)$ a decoder **D** (Sutskever et al., 2014), where **E** and **D** are single-layer RNNs with GRU cells (Chung et al., 2014). **E** is initialized with the dense vector of the source style $s$ ($s \in \{s_1, s_2\}$) and takes an input sequence $\mathbf{x}_s$ and outputs the latent representation of the input as $\mathbf{z_s} = E(x_s, s)$. **D** is initialized with the dense vector of the target style $s\prime$ ($s\prime \in \{s_1, s_2\}$) and $\mathbf{z}$, it then generates a sequence in the desired style $\widetilde{\mathbf{x}}_{s\prime}$. The style vector and input tokens vectors are initialized randomly and their embedding layers are trained throughout the training process.

If the source and the target styles are the same ($s = s\prime$), **E** and **D** form an auto-encoder model which is trained to reconstruct the input sequences by minimizing the reconstruction loss (equation 3.1) which is the cross-entropy between the input sequence $\mathbf{x}_s$ and its reconstructed output text $\widetilde{\mathbf{x}}_{\mathbf{s}}^{(rec)}$. $\theta_E$ and $\theta_D$ are the parameters of the encoder and decoder that are estimated during the training. If the source style and the target style are not the same ($s \neq s\prime$), **D** creates a style-shifted sequence $\widetilde{\mathbf{x}}_s^{(trf)}$ for the input $x_s$ which cannot used for computing the reconstruction loss since the training is done in an unsupervised manner[10].

$$\mathscr{L}_{rec}(\theta_E, \theta_D) \quad = \quad -\log \Pr(\widetilde{\mathbf{x}}_{s_1}^{(rec)} | \mathbf{x}_{s_1}) - \log \Pr(\widetilde{\mathbf{x}}_{s_2}^{(rec)} | \mathbf{x}_{s_2}) \tag{3.1}$$

[9]The code is released under "Apache-2.0 license"

[10]We do not use gold style-shifted texts corresponding to inputs while training even if they are available.

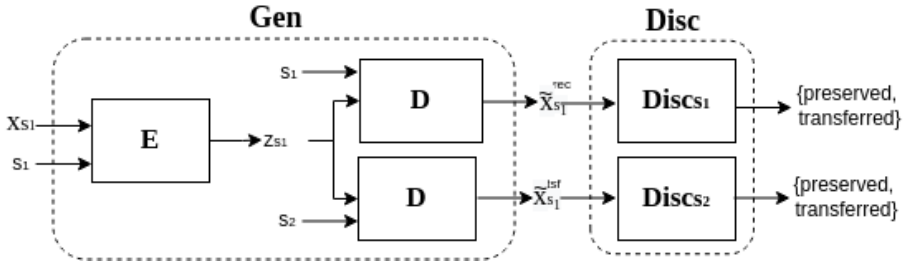


Figure 3.1 Adversarial TST RNN-based baseline model, *Gen*: Generator block, and *Disc*: Discriminator block.

During the training, at each training step, two inputs $\mathbf{x}_{s1}$ and $\mathbf{x}_{s2}$ (where $s_1 \neq s_2$) are processed in parallel. Firstly, the embedding representations of theses two sequences are created as the last state of **E**: $\mathbf{z_{s_1}} = \mathbf{E}(x_{s_1}, s_1)$ and $\mathbf{z_{s_2}} = \mathbf{E}(x_{s_2}, s_2)$. Then, given $\mathbf{x}_{s_1}$, a reconstructed sequence $\widetilde{\mathbf{x}}_{s_1}^{(rec)} = \mathbf{D}(z_{s1}, s_1)$ is created where **D** is teacher-forced by the tokens of the $\mathbf{x}_{s_1}$ and a style-shifted sequence $\widetilde{\mathbf{x}}_{s_1}^{(trf)} = \mathbf{D}(z_{s1}, s_2)$ is generated where **D** is self-fed by the soft distribution of the generated token in the previous step. Similarly, the corresponding reconstructed and style-shifted sequences are created for $\mathbf{x}_{s_2}$.

#### 3.2.1.2 Discriminator block

For each style, the discriminator block ***Disc*** contains a style-specific classifier $\mathbf{Disc_s}$ ($s \in \{s_1, s_2\}$) which is a single layer Feed-Forward network with a sigmoid output layer. $\mathbf{Disc_{s1}}$ takes as the input the decoder RNN hidden states corresponding to the reconstructed text with the style $s_1$ and the style-shifted sequence with the desired style $s_1$. Similarly, $\mathbf{Disc_{s2}}$ is fed with the decoder RNN hidden states corresponding to the reconstructed text with the style $s_2$ and the style-shifted sequence with the desired style $s_2$. $\mathbf{Disc_{s1}}$ and $\mathbf{Disc_{s1}}$ compute $Pr(\text{“}preserved\text{”})$ for each input, i.e. the probability that the input preserved its source style. Therefore, they are trained to assign reconstructed inputs with label 1 (“*preserved*”) and style-shifted inputs with label 0 (“$transferred$”). They are trained jointly with ***Gen*** by minimizing the respective equations 3.2 and 3.3, binary cross-entropy loss, where $\theta_{Disc_s}$ is

the parameters of the classifier for the style $s$ ($s \in \{s_1, s_2\}$).

$$\mathscr{L}_{Disc_{s1}}(\theta_{Disc_{s_1}}) = -\log(Disc_{s_1}(\widetilde{\mathbf{x}}^{rec}_{s_1})) - \log(1 - Disc_{s_1}(\widetilde{\mathbf{x}}^{trf}_{s_2})) \tag{3.2}$$

$$\mathscr{L}_{Disc_{s2}}(\theta_{Disc_{s_2}}) = -\log(Disc_{s_2}(\widetilde{\mathbf{x}}^{rec}_{s_2})) - \log(1 - Disc_{s_2}(\widetilde{\mathbf{x}}^{trf}_{s_1})) \tag{3.3}$$

#### 3.2.1.3 Adversarial training

The training of the TST model is done in an adversarial manner where ***Disc*** aims at detecting the style-shifted segments and labelling them as 0 ("$transferred$") and the training objective for ***Gen*** is to create style-shifted sequences in the desired style such that it fools the discriminator into labelling them as 1 ("preserved"). This leads to maximizing the adversarial loss which is computed in equation 3.4 (equation 3.5 is the symmetrical equation for $\mathbf{Disc_{s_2}}$).

$$\mathscr{L}_{adv,s_1} = \log(1 - Disc_{s_1}(\widetilde{\mathbf{x}}^{(trf)}_{s_2})) \tag{3.4}$$

$$\mathscr{L}_{adv,s_2} = \log(1 - Disc_{s_2}(\widetilde{\mathbf{x}}^{(trf)}_{s_1})) \tag{3.5}$$

The total loss (equation 3.6) is the summation of the reconstruction and adversarial losses. If ***Disc*** detects that the textual segment is style-shifted $\mathscr{L}_{adv,s}$, and as a result $\mathscr{L}_{total}$ increases which penalizes the whole training process. Therefore, ***Gen*** aims at minimizing the $\mathscr{L}_{adv,s}$ which means increasing the chances of the style-shifted sequences being detected as "preserved" by $\mathbf{Disc_s}$. ***Disc*** and ***Gen*** are trained jointly from scratch. Backpropagation is done for ***Gen*** by using equation 3.6 to update $\theta_E$ and $\theta_D$ and for ***Disc*** by using the equations 3.2 and 3.3 to estimate $\theta_{D_s}$ ($s \in \{s_1, s_2\}$).

$$\mathscr{L}_{total}(\theta_E, \theta_D) = \mathscr{L}_{rec} + \mathscr{L}_{adv,s_1} + \mathscr{L}_{adv,s_2} \tag{3.6}$$

**Algorithm 1 : Adversarial training of RNN-based TST model**

**Input:** Generator block *Gen* ($\theta_E, \theta_D$), discriminators $Disc_{s1}$ ($\theta_{Disc_{s1}}$) and $Disc_{s2}$ ($\theta_{Disc_{s2}}$) and any two corpora $X_{s_1}$ and $X_{s_2}$ which have the same content distribution but different styles $s_1$ and $s_2$ ($s_1 \neq s_2$). This means that for instance, the two datasets should be both restaurant reviews but they should have different sentiments.

**1.** Sampling two mini-batches with the size $k$ from the sets $X_{s_1}$ and $X_{s_2}$ (setting $k = 1$ for the sake of simplicity).
**2.** Processing the two mini-batches in parallel, i.e. for each of the sequences $x_{s_1} \in X_{s_1}$, and $x_{s_2} \in X_{s_2}$, ***Gen*** generates a reconstructed and a style-shifted sequence for each input:

- For $x_1$: $\widetilde{\mathbf{x}}_{s_1}^{(rec)} = Gen(x_{s_1}, s_1)$
  $\widetilde{\mathbf{x}}_{s_1}^{(trf)} = Gen(x_{s_1}, s_2)$
- For $x_2$: $\widetilde{\mathbf{x}}_{s_2}^{(rec)} = Gen(x_{s_2}, s_2)$
  $\widetilde{\mathbf{x}}_{s_2}^{(trf)} = Gen(x_{s_2}, s_1)$

**3.** Computing $\mathscr{L}_{rec}$ by equation 3.1 using reconstructed sequences.
**4.** Computing $\mathscr{L}_{Disc_{s1}}$ and $\mathscr{L}_{Disc_{s2}}$ by equation 3.2 equation 3.3 and perform gradient decent to update $\theta_{Disc1}$ and $\theta_{Disc2}$ using both reconstructed and style-shifted sequences.
**5.** Considering equations 3.2 and 3.3, if the condition $\mathscr{L}_{Disc} < 1.2$ (a pre-set threshold which we set to 1.2 following Shen et al. (2017)) holds:
**-** Computing $\mathscr{L}_{adv}$ by equations 3.4 and 3.5, then, carrying out the gradient decent to update $\theta_{Gen}$ by using the equation 3.6.
**-** Otherwise: performing the backpropagation for the *Gen* by only using the $\mathscr{L}_{rec}$ (equation 3.1).
**6.** Repeating the steps 1, 2 , 3, 4 and 5 for the number of epochs (a hyperparameter set to 20 here).
**7.** Selecting the model with lowest total loss (equation 3.6) as the best model.

### 3.2.2 Training

Firstly, if the sizes of training sets of the styles $s_1$ and $s_2$ differ, while reading the data upsampling without repetition would be done to equalize the size of the two sets. The same holds for the development set, i.e. if the binary-styled development sets are different in size, as a preprocessing step, upsampling without repetition would be done for the set with the smaller size before the training starts. Then, the training process starts which follows the steps summarized in the training algorithm: *Algorithm 1: Adversarial training of RNN-based TST model*.

In the remaining chapters, all RNN-based models (i.e. the presented baseline system as well as all its variants that we well introduce in chapters 4.1 and 5) follow this training procedure. When small differences exist, they will be detailed.

### 3.2.3 Transformer-based TST baseline model

The baseline transformer-based (T-based) adversarial $TST$ model that is used throughout this research is similar to the model proposed by Dai et al. (2019b). This architecture is composed of a generator block and a discriminator block. The generator block contains a T-based encoder and a T-based decoder. The discriminator block is a binary classifier which is trained together with the generator block. The training of the generator block which is responsible for rewriting the input text in a desired style is done in an unsupervised manner by applying adversarial techniques and receiving style signals from the discriminator block. We explain the TST encoder-decoder network and the discriminator block, as well as the training steps in the following sections.

#### 3.2.3.1 Generator block

The generator block (**Gen**) of the baseline T-based model is a sequence-to-sequence encoder-decoder framework where both encoder (**E**) and decoder (**D**) are transformer architectures similar to the model introduced by Vaswani et al. (2017).

**Encoder E** **E** contains an embedding layer and 4 stacks of transformers (figure 3.2). Each stack of transformer is identical and consists of a fully connected self-attention, a fully connected point-wise Feed-Forward layer, and 2 residual normalization layers (see section 2.1.4.3).

**E** takes a sequence $x_{s1}$ of the length $T$ and a desired style $s_2$ as the input where the desired style is taken an extra token of the input sentence. It first projects the input tokens to one

embedding matrix and desired styles to another embedding matrix where token embedding vectors and style embedding vectors are both initialized randomly and the model learns not only the context but also their positional information[11] of the tokens as well as dense representations of different styles (the positional encoding is not used for the style tokens).

Then, the sequence of the vector representation of the desired style and the embedded tokens are fed to the first stack of the **E** where the output of each layer is first normalized by a residual layer, and then is fed as the input to the next layer. This means that the residual mechanism of a layer normalizes the addition of the inputs and outputs of that layer and feeds the result to the next layer. The final layer of ***E*** generates a sequence of latent token representations: $z = (z_0, z_1, ..., z_T)$ where $z_0$ is the dense representation of the desired style. The source style of inputs $s_1$ and the given desired styles $s_2$ can be the same or different depending on whether the goal is to generate a style-shifted text ($s_1 \neq s_2$) or a reconstructed sequence ($s_1 = s_2$).

**Decoder D** **D**, similar to **E**, starts its processing by projecting the input tokens (if gold output tokens are available which is the case when reconstructing the text) through contextual and positional embedding layers. The embedding layer is followed by 4 stacks of transformer where each stack includes an attention layer, a fully connected point-wise Feed-Forward layer, and a normalization layer. Fully connected point-wise Feed-Forward, as well as the 3 residual normalization layers in each stack of **D** perform similarly to those of the residual layers of the stacks of **E**.

The attention mechanism of the stacks of **D** consists of not only a fully connected self-attention layer which implements a self-attention mechanism similar to that of the **E**, but also a fully connected encoder-decoder attention layer. This layer, as figure 3.2 illustrates, takes the sequence of vector representations of the input tokens $z$ (created by **E**) together with the

[11] Following Vaswani et al. (2017), fixed positional embeddings are applied here.

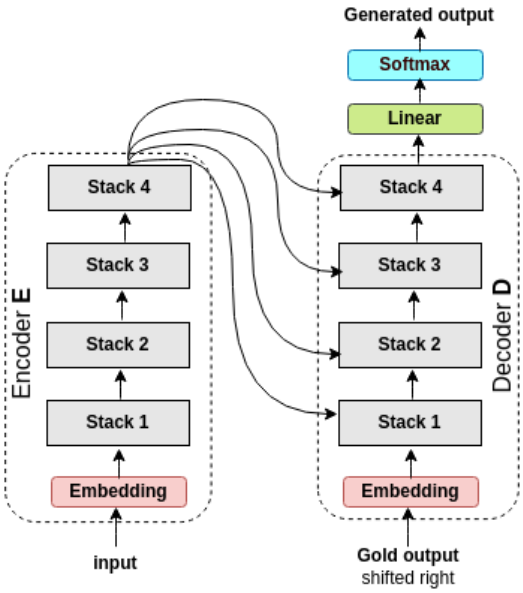


Figure 3.2 Generator block **Gen** of the T-based model, **E** and **D** consists of 4 stacks of transformer and the output of the last stack of **E** is fed to each stack of **D**.

output of the self-attention layer and computes the weights corresponding to the input tokens at each time step.

The last component of the **D** subnetwork is the projection layer (figure 3.2) which generates the output token given the outputs of the last layer of the **D**-stack.

#### 3.2.3.2 Discriminator block

The discriminator block (***Disc***) is a binary classifier which is trained together with the ***Gen***. ***Disc***, similar to some previous work such as Radford et al. (2018b) and Devlin et al. (2018), consists of a sequence of a transformer and a classifier (figure 3.3). The classifier is a Feed-Forward network with a single hidden layer and a softmax output layer. The transformer architecture of ***Disc*** is identical to that of ***E*** (section 3.2.3.1) containing an embedding layer followed by 4 stacks of transformers where each stack has a fully connected self-attention followed by a residual normalization layer, as well as a fully connected point-wise Feed-Forward layer followed by another normalization layer.

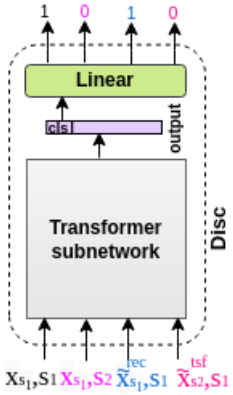


Figure 3.3 Discriminator block ***Disc*** is trained to label the original and reconstructed text having the source style as True (1), style-shifted sequences having the reverse style as False (0) and original sequences having the reverse style as False (0).

As an input, ***Disc*** takes a text and a style, and aims at detecting whether or not the original style of the text matches the given style, i.e. given a pair of $(x, s)$, it computes the probability of whether the original style of $x$ is $s$. ***Disc*** learns to return true for an original text and its original style and a reconstructed text and its original style and false for a style-shifted text and its desired style (figure 3.3). Specifically, it is trained to label input pairs of either a source or reconstructed sequence and the source style as true, 1: $(\mathbf{x}_{s_1}, s_1)$ and $(\widetilde{\mathbf{x}}_{s_1}^{(rec)}, s_1)$, and style-shifted sequences and their desired style as well as source sequences and their reverse styles as false, 0: $(\widetilde{\mathbf{x}}_{s_1}^{(trf)}, s_2)$ and $(x_{s1}, s_2)$.

To do this, ***Disc*** reads a textual input, projects the tokens to their positional and contextual embedding vectors. Then, it feeds the first transformer stack with the embedding representation of the input tokens which is augmented by the dense vector of the style (either source or reverse) and a special token `<cls>`. Following some previous work (Radford et al., 2018b; Kenton and Toutanova, 2019), only the vector in the position corresponding to the `<cls>` token of the output of the transformer network of the ***Disc*** is fed to its linear classification layer. The softmax probabilities of this linear layer are considered as the outputs of the ***Disc***. During training, ***Disc*** uses these outputs to minimize $\mathscr{L}_{Disc}$ which is defined in equation

3.7[12] as the binary cross-entropy over the two classes where $s_1 \neq s_2$.

$$\begin{aligned} \mathscr{L}_{Disc} &= -\log(Disc(\widetilde{\mathbf{x}}_{s_1}^{(rec)}, s_1)) \\ &\quad -\log(1 - Disc(\widetilde{\mathbf{x}}_{s_1}^{(trf)}, s_2)) \end{aligned} \tag{3.7}$$

#### 3.2.3.3 Adversarial training

In each step of training, two sequences of $\mathbf{x}_{s1}$ and $\mathbf{x}_{s2}$ are processed in parallel where $\mathbf{x}_{s1} \in X_1$ and $\mathbf{x}_{s2} \in X_2$[13]. $X_1$ and $X_2$ are two sets with the same content distribution and different styles. Taking $\mathbf{x}_{s1}$ and $\mathbf{x}_{s2}$ as the input, and the desired output style, ***Gen*** generates the following 4 sequences, i.e. a reconstructed and a style-shifted sequence for each input text:

- $\widetilde{\mathbf{x}}_{s_1}^{(rec)} = Gen(x_{s_1}, s_1)$
- $\widetilde{\mathbf{x}}_{s_1}^{(trf)} = Gen(x_{s_1}, s_2)$
- $\widetilde{\mathbf{x}}_{s_2}^{(rec)} = Gen(x_{s_2}, s_2)$
- $\widetilde{\mathbf{x}}_{s_2}^{(trf)} = Gen(x_{s_2}, s_1)$

To motivate the TST model to preserve the information of the input text we define the reconstruction loss as follows.

**Reconstruction loss** When the input style and the output desired style are the same, the model simply aims at reconstructing the given text and, in practice, it acts similarly to an auto-encoder. To enable the model to rewrite the output, we define a self-reconstruction loss $\mathscr{L}_{self_{rec}}$. To calculate the ($\mathscr{L}_{self_{rec}}$), the reconstructed and input text are used as the generated and gold tokens and the negative log probability of each input sequence $\mathbf{x}$ and its corresponding reconstructed sequence $\widetilde{\mathbf{x}}$ is minimized using equation 3.8 during the training.

[12]The equation is similarly computed for the input pairs of ($\mathbf{x}_{s_1}$, $s_1$) and ($\mathbf{x}_{s_1}$, $s_2$).

[13]For the sake of simplicity, we consider the batch-size = 1.

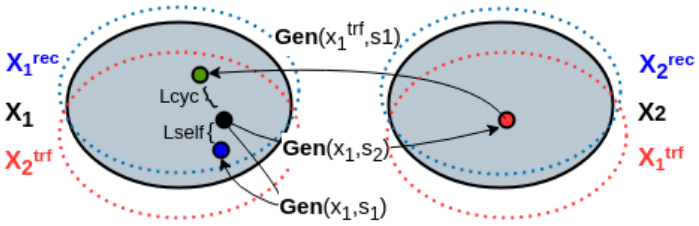


Figure 3.4 Computing the cycle loss

$$\mathscr{L}_{self_{rec}} = -\log \Pr(\widetilde{\mathbf{x}}_s^{(rec)} = \mathbf{x_s} | \mathbf{x_s}, \mathbf{s}) \tag{3.8}$$

On the other hand, when the input style and the output desired style differ, the TST model aims at generating style-shifted outputs and due to using non-parallel data, there is no access to gold sequences for style-shifted outputs. Therefore, equation 3.8 cannot be used to compute the $\mathscr{L}_{self_{rec}}$ for style-shifted text. To better encourage the model to preserve the non-stylistic information of the input, we define $\mathscr{L}_{cycle_{rec}}$. To do so, given the input $x_{s1}$, the model does a cycle of generating style-shifted text which leads to reconstructing the input and creating $\widetilde{\mathbf{x}}_{s_1}^{(rec)\prime}$. To be more precise, as figure 3.4 shows, the model follows these two steps ($s_1 \neq s_2$):

1. Feeding ***Gen*** with the desired style $s_2$ and $x_{s1}$ to generate $\widetilde{\mathbf{x}}_{s_1}^{(trf)}$.

   $$\widetilde{\mathbf{x}}_{s_1}^{(trf)} = Gen(x_{s_1}, s_2)$$

2. Feeding ***Gen*** with the desired target style $s_1$ and $\widetilde{\mathbf{x}}_{s_1}^{(trf)}$ which has the source style $s_2$ to generate the style-shifted form of $\widetilde{\mathbf{x}}_{s_1}^{(trf)}$ which is the reconstructed form of $x_{s1}$.

   $$\widetilde{\mathbf{x}}_{s_1}^{(rec)\prime} = (\widetilde{\mathbf{x}}_{s_1}^{(trf)})_{s_2}^{(trf)} = Gen(\widetilde{\mathbf{x}}_{s_1}^{(trf)}, s_1)$$

Doing this cycle of generation, $\mathscr{L}_{cycle_{rec}}$ can be computed similar to $\mathscr{L}_{self_{rec}}$ (equation 3.8) by minimizing the negative log probability of each input sequence $x$ and its corresponding reconstructed sequence $\widetilde{\mathbf{x}}_s^{(rec)\prime}$.

$$\mathscr{L}_{cycle_{rec}} = -\log \Pr(\widetilde{x}_s^{(rec)\prime} = \mathbf{x_s} | \widetilde{\mathbf{x}}_\mathbf{s}^{(\mathbf{trf})}, \mathbf{s}) \tag{3.9}$$

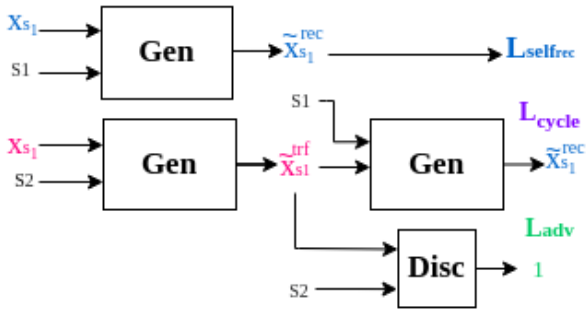


Figure 3.5 The schema of the T-based TST baseline model.

The total reconstruction loss $\mathscr{L}_{total_{rec}}$ of the model is the weighted summation of the self-reconstruction loss $\mathscr{L}_{self_{rec}}$ and the cycle-reconstruction loss $\mathscr{L}_{cycle_{rec}}$ and is computed using the equation 3.10.

$$\mathscr{L}_{total_{rec}} = \alpha \mathscr{L}_{self_{rec}} + \beta \mathscr{L}_{cycle_{rec}} \tag{3.10}$$

**Adversarial loss** To compensate for the lack of parallel data while training the T-based baseline TST model adversarial techniques are applied to control the generation of the text to include the desired style. ***Disc*** as the key component of the adversarial training competes with ***Gen*** in distinguishing the style-shifted text from the reconstructed text, while ***Gen*** attempts to improve the generation so that style-shifted outputs cannot be categorized. To enable this competition, an adversarial loss is defined, equation 3.11 where $s_1 \neq s_2$.

$$\mathscr{L}_{adv} = -\log(Disc(\widetilde{\mathbf{x}}_{s_1}^{(trf)}, s_2)) \tag{3.11}$$

If ***Disc*** detects that the style of the input $\widetilde{\mathbf{x}}_{s_1}^{(trf)}$ is shifted to $s_2$, it labels $(\widetilde{\mathbf{x}}_{s_1}^{(trf)}, s_2)$ as 0. This leads to the increase of adversarial loss and, as a result, the increase of the total loss which is the summation of adversarial loss and reconstruction loss (equation 3.12). Therefore, ***Disc*** detecting the style-shifted text leads to penalizing the training process which encourages ***Gen*** to improve the generation of the style-shifted text. During training, adversarial loss is

minimized together with the total loss (equation 3.12).

$$\mathscr{L}_{total} = \mathscr{L}_{total_{rec}} + \mathscr{L}_{adv} \tag{3.12}$$

To train the T-based baseline TST model, firstly, if the training sets of the styles $s_1$ and $s_2$ have different sizes, upsampling without repetition is done. Also, if the size of the binary-styled development set differs, downsampling is done to the size of the development set with the smaller size. After this preprocessing step the training process starts following the steps summarized in the training algorithm: *Algorithm 2: Adversarial training of T-based TST baseline model*.

### 3.2.4 Experimental setup

This section reports the parameters of the baseline frameworks used throughout the experiments of the later chapters. The parameters used in RNN-based baseline model are described in table 3.8 and the parameters used in T-based baseline model are described in table 3.9.

Table 3.8 Hyperparameters of the RNN-based TST baseline models

| **Parameter Name** | **Description** | **Value** |
|---|---|---|
| ***E*** type | Uni-directional GRU cells | - |
| ***D*** type | Uni-directional GRU cells | - |
| ***E*** Depth | Number of layers of ***E*** RNN | 1 |
| ***D*** Depth | Number of layers for ***D*** RNN | 1 |
| ***E*** cell size | GRU hidden unit size of ***E*** | 700 |
| ***D*** cell size | GRU hidden unit size of ***D*** | 700 |
| Style size | Dense style vector size | 200 |
| Continued on next page | | |

**Table 3.8 – continued from previous page**

| Parameter Name | Description | Value |
|---|---|---|
| Embedding size | Embedding size of tokens | 100 |
| Pre-trained model | Pre-trained embedding to initialize tokens | GloVe |
| Batch size | - | 64 |
| Epoch number | - | 20 |
| ***Disc*** type | Binary style-specific Text CNN classifier | - |
| Optimizer | Optimizer for both ***Gen*** & ***Disc*** | Adam |
| ***Gen*** Learning rate | - | 0.0005 |
| ***Disc*** Learning rate | - | 0.0005 |
| Min frequency | Min frequency of tokens to appear in vocabulary dictionary | 5 |

Table 3.9 Hyperparameters of the T-based TST baseline model

| Parameter Name | Description | Value |
|---|---|---|
| ***Gen*** type | An ***E*** transformer & a ***D*** transformer | - |
| ***E*** Depth | Number of stacks of ***E*** transformer | 4 |
| ***D*** Depth | Number of layers for ***D*** transformer | 4 |
| ***E*** attention head | Number of attention heads of ***E*** | 4 |
| ***D*** attention head | Number of attention heads of ***D*** | 4 |
| Style size | Dense style vector size | 256 |
| Token Embedding | ***Gen*** positional embedding size | 256 |
| Position Embedding | ***Gen*** positional embedding size | 256 |

Continued on next page

**Table 3.9 – continued from previous page**

| Parameter Name | Description | Value |
|---|---|---|
| ***Gen*** hidden size | Dense style vector size | 256 |
| ***Disc*** type | A sequence of a transformer and a classifier (a feed-forward network) | - - |
| ***Disc*** Depth | Number of stacks of ***Disc*** transformer | 4 |
| ***Disc*** attention head | Number of attention heads of ***Disc*** | 4 |
| ***Disc*** hidden size | Dense style vector size | 256 |
| Token Embedding | ***Disc*** token embedding size | 256 |
| Position Embedding | ***Disc*** positional embedding size | 256 |
| `<cls>` size | The size of `<cls>` token of the ***Disc*** | 256 |
| Optimizer type | Optimizer type for both ***Disc*** & ***Gen*** | Adam |
| ***Gen*** Learning rate | - | 0.0001 |
| ***Disc*** Learning rate | - | 0.0001 |
| Pre-training iteration ($n_p$) | Number of updates of ***Gen*** in pre-training step | 500 |
| ***Gen*** iteration ($n_g$) | Number of ***Gen*** updates per training iteration | 5 |
| ***Disc*** iteration ($n_d$) | Number of ***Disc*** update per training iteration | 10 |
| Evaluation step ($n_{eval}$) | Number of steps after which the model performance is evaluated on development data | 25 |
| Batch size | - | 64 |
| Epoch number | - | 20 |
| Min frequency | Min frequency of tokens to appear in vocabulary dictionary | 5 |

**Algorithm 2: Adversarial training of T-based TST baseline model**

**Input:** ***Gen***$(\theta_E, \theta_D)$, ***Disc*** $(\theta_{Disc})$ and any two corpora $X_{s_1}$ and $X_{s_2}$ which have the same content distribution but different styles of $s_1$ and $s_2$ $(s_1 \neq s_2)$.

**1.** Sampling two batches with the size $k$ from the sets $X_{s_1}$ and $X_{s_2}$ (setting $k = 1$ for the sake of simplicity).
**2.** Processing the two mini-batches in parallel, i.e. for each of the sequences $x_{s_1} \in X_{s_1}$, and $x_{s_2} \in X_{s_2}$, ***Gen*** generates a reconstructed and a style-shifted sequence for each input:

- For $x_1$: $\widetilde{\mathbf{x}}_{s_1}^{(rec)} = G(x_{s_1}, s_1)$
  $\widetilde{\mathbf{x}}_{s_1}^{(trf)} = G(x_{s_1}, s_2)$
- For $x_2$: $\widetilde{\mathbf{x}}_{s_2}^{(rec)} = G(x_{s_2}, s_2)$
  $\widetilde{\mathbf{x}}_{s_2}^{(trf)} = G(x_{s_2}, s_1)$

**3.** Computing the self-reconstruction loss using the equation 3.8.
**4.** Computing the discriminator loss using the equation 3.7.
**5.** Computing the cycle-reconstruction loss using the equation 3.9 and adversarial loss using the equation 3.11.
**6.** Pre-training the model by repeating step 1 and 2 and 3, and performing gradient decent to update $\theta_E$, $\theta_D$ for $n_p$ times.
**7.** Training ***Disc*** by repeating step 1 and 2 and 4, and performing gradient decent to update $\theta_{Disc}$ for $n_d$ times.
**8.** Training ***Gen*** by repeating steps 1 and 2, and then the following steps for $n_g$ times:

- First do step 3, and perform gradient decent to update $\theta_E$, $\theta_D$.
- Second do step 5, and perform gradient decent to update $\theta_E$, $\theta_D$.

**9.** For all the epochs (20 here), repeating steps 7, and 8 for all the batches in the training set and selecting the model with lowest total loss (equation 3.12) as the best model in each evaluation step ($n_{eval}$).

* $n_p$, $n_d$, $n_g$ and $n_{eval}$ are hyperparameters that their values are specified in table 3.9.
** Neither training stop-condition nor model selection strategy was stated in the training steps of the T-based model (Dai et al., 2019b); therefore, to train this model, we set the conditions described in step **9**.
*** Each evaluation step is after doing steps **8** and **9** for $n_{eval}$ times, i.e. after iterating over $(n_{eval} * (n_d + n_g))$ number of batches during training.

## 3.3 Evaluation methodology

The evaluation methodology we use for our experiments considers three dimensions: content preservation, style transfer strength and fluency. We believe that, taken together, these evaluation aspects and methods provide a comprehensive evaluation methodology for textual style transfer. We further confirm this methodology through a human evaluation.

### 3.3.1 Automatic evaluation

This section describes the automatic metrics used to compute the performance of the TST models in the three aspects of style transfer power (section 3.3.1.1), content preservation power (section 3.3.1.3) and fluency (section 3.3.1.2).

#### 3.3.1.1 Style-shift power (*SSP*)

*SSP* investigates how well a TST model performs in shifting the style of the inputs. To compute *SSP*, we followed previous work (Fu et al., 2018a; Li et al., 2018a; Leeftink and Spanakis, 2019; Singh and Palod, 2018; Prabhumoye et al., 2018; Shen et al., 2017; John et al., 2019; Hu et al., 2017a) and trained style classifiers which predict the probability of the style-shifted text to have the desired style. If these classifiers label a generated style-shifted sequence with the desired style, it shows that the TST model has shifted the style of the sequence successfully. Therefore, the percentage of the style-shifted text which are labelled with the desired style by this classifier signifies the power of the TST model in shifting the textual style (SSP).

Throughout our experiments, similar to the approach taken by some previous research, such as Shen et al. (2017), we use the TextCNN model proposed by Kim (2014) as style classifier. To measure the *SSP* of TST models, for each dataset, we train a separate style classifier for on the same training data.

#### 3.3.1.2 Fluency (*PPL*)

Following previous research (Zhao et al., 2018; John et al., 2019), we examine the fluency of style-shifted sequences in terms of their grammatical correctness by considering perplexity (*PPL*) and compute it using pre-trained language models.

The *PPL* of an unseen textual sequence $x = [w_1, ..., w_T]$) is its inverse probability, $Pr(w_1 w_2 ... w_N)$, normalized by the number of tokens (equation 3.13) (Shi, 2017).

$$PPL(X) = Pr(w_1 w_2 ... w_N)^{\frac{1}{N}} = \sqrt[N]{\frac{1}{Pr(w_1 w_2 ... w_N)}} \tag{3.13}$$

Language models calculate the probability of the given text using the chain rule.

$$Pr(w_1 w_2 ... w_N) = \prod_{t=1}^{N} Pr(w_t | w_1^{t-1}) \tag{3.14}$$

The inversion while computing *PPL* means that minimizing *PPL* results in maximizing probability, i.e. lower *PPL* scores represent higher fluency in the generated texts of the models. In other words, the fluency score of a sequence is negatively related to its *PPL*.

To compute *PPL* of the generated text of the TST models, for each dataset, we train a separate RNN-based language model consisting of a single-layer RNN with the uni-directional GRU cell (Chung et al., 2014). The tokens are initialized by embedding vectors using 100-dimensional pre-trained embedding GloVe model (Pennington et al., 2014). The *PPL* reported for each model is computed as the average score of the *PPL* of each style-shifted output over the test data of a corpus.

#### 3.3.1.3 Content preservation power (*CPP*)

This aspect of evaluation focuses on how well a style-shifted sequence maintains the content of the input sequence. Different approaches have been suggested in the literature to compare two sequences and measure their semantic similarity. These methods can be categorized as

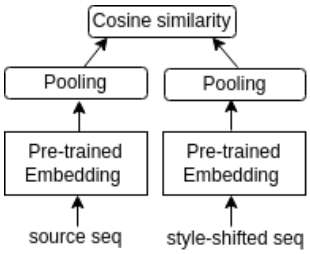


Figure 3.6 Computing *CPP* scores using embedding-based metrics

embedding-based or unigram-based strategies. We use the following metrics to investigate the performance of the models in terms of the content maintenance power.

**Embedding-based metrics** We follow the approach proposed by Fu et al. (2018a) in computing the content maintenance of a given style-transferred sequence as the cosine similarity between its embedding representation and embedding vector of its corresponding source input. To generate embedding vectors of the sequences, a pre-trained embedding model is used to map each token to its embedding. Then, a pooling layer is applied over these token embeddings (figure 3.6). Here, two types of pooling layers, average and min-max-average techniques, are used which we explain more about in the following sections. The *CPP* score of a model is the average of scores computed for all style-shifted outputs created by that model. In this research, as pre-trained embedding models, we use both GloVe (Pennington et al., 2014) and SBERT models (Reimers and Gurevych, 2019). We refer to these cosine similarity based metrics as GloVe-based and SBERT-based.

- *GloVe-based* We use a 100-dimensional GloVe model and map the tokens of sequences to their embeddings, represented as $e_i \in \mathbb{R}^{100}$ $(1 \leq i \leq N)$. The embedding vector of a given sequence with the size $N$ is then created as the concatenation of the following pooling vectors of its token embeddings: $min = \left(\min_{1\leq i\leq N} e_{i,j}\right)_{1\leq j\leq 100}$, $mean = \left(\frac{\sum_{1\leq i\leq N} e_{i,j}}{N}\right)_{1\leq j\leq 100}$, and $max = \left(\max_{1\leq j\leq 100} e_{i,j}\right)_{1\leq i\leq N}$.

- *SBERT-based metric* We use a pre-trained SBERT model[14] which creates embedding representation for sequences by firstly mapping the tokens to their embedding vectors (with the size 768), i.e. given a sequence with the size $N$, tokens are created as $e_i \in \mathbb{R}^{768}$ $(1 \leq i \leq N)$. Then, it takes the average of these embeddings $mean = \left(\frac{\sum_{1 \leq i \leq N} e_{i,j}}{N}\right)_{1 \leq j \leq 768}$ as the sequence vector.

**Word Mover's Distance (WMD)** is an embedding-based method which is a special case of the Earth Mover's Distance (Rubner et al., 2000) and has been used in some previous style transfer research to compute the *CPP*, such as Yamshchikov et al. (2020). It calculates the distance of the sequences in the embedding space. $WMD$ uses a distance technique and matches the tokens of the two sequences by measuring their distance in the semantic space, i.e. the tokens that are closer are matched with each other. Then, it calculates the distance score of the two sequences as the average of the distances of their matched tokens (Kusner et al., 2015). Lower distance scores in $WMD$ demonstrate higher semantic resemblance.

To compute $WMD$, we first mapped the tokens of the style-shifted and input sequences to their embedding representation using a pre-trained 300-dimensional Word2Vec model (Shivakumar and Georgiou, 2019).

**Word overlap (WO)** is a unigram-based method proposed by (John et al., 2019) which computes the unigram overlap of two sequences. For instance, it calculates the ratio of the unigram overlap of the tokens of a given input sequence $x$ and the tokens of its corresponding generated output text $\widetilde{x}$ and the total number of the tokens of the two sequences (equation 3.15).

$$WO = \frac{count(x \cap \widetilde{x})}{count(x \cup \widetilde{x})} \tag{3.15}$$

[14] https://huggingface.co/cross-encoder/cross-encoder/stsb-TinyBERT-L-4

There are other unigram-based metrics which are used in *NLG* evaluation. One of the widely applied unigram-based metric is BLEU (Papineni et al., 2002). BLEU score (considering unigrams) for a given pair of input and style-shifted text is computed as the ratio of the unigram overlap of style-shifted tokens and input tokens and total number of tokens of the style-shifted text. Following some previous TST research such as John et al. (2019), we use this metric throughout the experiments of this thesis and report the $WO$ value for a set of style-shifted text is the average of $WO$ scores computed between the sequence pairs of that set and their corresponding reference set where the stop words are removed from the sequences as a preprocessing step[15].

#### 3.3.1.4 Upper Bounds and Lower Bounds of the automatic evaluation metrics

Given an evaluation dimension, the lower bound score of a metric can be assumed as the scores computed using some of the worst possible outputs. Similarly, the upper bound score of a related metric reflects how ideally models should perform. These scores can be computed using gold data as the model outputs.

To better interpret how well TST models perform in maintaining the content of the inputs, shifting their style and generating fluent text, we compute the lower bound and upper bound scores of evaluation metrics which were introduced in section 3.3.

**SSP:** To measure the lower bound of *SSP* for a given corpus, we consider the test set of that corpus and take the set having the style $s_1$ as the style-shifted set with desired style $s_2$ and the test set of style $s_2$ as the style-shifted set with desired style $s_1$ ($s_2 \neq s_1$). The idea being that the worst style-shifted sequence for a desired style can be the gold text of the opposite style. Similarly, the best style-shifted sequences for a desired style can be the gold

[15]We also computed $WO$ score while keeping the stop words and the observed that the two sets of scores ranked the models similarly.

text of the same style. Therefore, to compute upper bounds, we also use the test data where the test set of style $s$ is considered as the style-shifted data with the desired style $s$.

To calculate the scores, for each corpus, we use the related classifier trained for computing the *SSP* (explained in section 3.3.1.1) and report the percentage of the style-shifted text which are labelled with the desired style.

**PPL:** To compute the lower bound of fluency (*PPL*) of each dataset, first, we shuffle the tokens of the sequences of test data. Then, we compute *PPL* of the shuffled test data as the average of *PPL* scores of sequences using the language model which is trained over the given dataset.

The upper bound *PPL* scores are computed as the perplexity that the language model trained over a dataset measures for the test set of that dataset.

**CPP:** To compute the lower bound of *CPP*, for each metric and data set, we calculate *CPP* score taking the sequences of test data and randomly selected sequences from train set of the given dataset. Then we take the average of *CPP* scores of all the random sequence pairs as the lower bound of that *CPP* metric.

The upper bound of *CPP* is computed as for all metrics of $WMD$, $WO$, $SBERT$-based and *GloVe*-based. It is computed by comparing each set with itself which results in the upper bound score 1 for all the metrics.

### 3.3.2 Human evaluation

Human evaluation tests are advantageous to be used in various NLP tasks. Firstly, due to the limitations of some of the automatic techniques, they can be applied to verify the automatic evaluation methodology. Some of these limitations in the scope of TST can be listed as follows. First, the *SSP* scores reported for the models using pre-trained classifiers (section 3.3.1.1) are affected by how well these classifiers are trained. Moreover, to compute

| | | Evaluation metrics | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | CPP | | | | SSP | | PPLX | |
| | | SBERT | GloVe | WMD | WO | | | | |
| | | **LB** | **LB** | **LB** | **LB** | **LB** | **UB** | **LB** | **UB** |
| **Data** | Yelp-*small* | 0.0939 | 0.86 | 1.16 | 0.0101 | 3.00 | 97.20 | 2052 | 43.77 |
| | Yelp-*large* | 0.0692 | 0.84 | 1.161 | 0.0103 | 2.53 | 97.47 | 2081 | 44.81 |
| | GYAFC-$v_2$ | 0.0672 | 0.87 | 1.122 | 0.0045 | 13.89 | 77.94 | 1226 | 74.21 |

Table 3.10 Lower Bound *LB* and Upper Bound *UB* scores of evaluation metrics across different datasets. The *UB* for all *CPP* metrics is 1 by comparing each file with itself. The higher values show better performance for all *CPP* metrics except for *WMD* and fluency *PPLX*.

the *CPP* of the TST models, the source and style-shifted sequences have different styles. This can affect the *CPP* scores assigned to these pairs due to the probable overlap between these two textual elements. Finally, the fluency of the TST models using the pre-trained language models (section 3.3.1.2) is firstly biased towards the data, i.e. if the gold data is not well structured grammatically, the model cannot be trained well. Secondly, the pre-trained language models are biased towards shorter text (Jin et al., 2022). Furthermore, manual evaluation techniques can be applied in the case of the lack of automatic evaluation techniques. For instance, in the scope of TST, which is a multi-dimensional task, there is lack of a comprehensive evaluation metric to do an overall ranking of the TST systems. Human tests can be conducted here so that annotators judge the style-shifted outputs of the TST frameworks taking the three aspects of the task into consideration.

However, the main drawback of evaluating the models manually is that this task is resource-consuming in terms of both time and finance. Moreover, the results of manual tests cannot be easily compared across different studies, since they are highly affected by the evaluators. Finally, the test can be difficult and ambiguous which can lead to low agreement between the judges reducing the validity of the test results.

Different strategies can be implemented while designing human evaluation tests. Broadly, the techniques can be categorized into groups of comparison-based tests as well as scoring-

based tests (Jin et al., 2022). In the former technique, the outputs of the models (at least two models) should be compared and ranked by testers. In the latter strategy, judges are asked to provide labels or scores directly to one output at a time. In this thesis, we use both of the strategies based on the requirements of the tests. For instance, based on whether there is a need for doing system comparison or assessing the intrinsic quality of the system outputs, comparison-based or scoring-based strategies were applied, respectively.

In this thesis, we strike a balance between the advantages of human evaluations and the difficulties in terms of time and financial limitations for the work by using human evaluation tests to verify the automatic evaluation framework we use in our experiments, rather than to test the specific performance of the systems. Consequently, in chapter 4 (section 4.2.1), we conduct human evaluations to verify the proposed automatic evaluation methodology by showing that human evaluators and automatic metrics rank the models similarly across the three evaluation dimensions covered by the automatic metrics. Then, in the experiments reported in the later chapters (5 & 6) we use the verified automatic evaluation methodology to study the performance of the TST models.

## 3.4 Summary

The current chapter firstly described the data used to perform the experiments during my thesis. Then, it focused on the modelling approach which is taken by all the TST frameworks in this research and it proceeded by introducing the RNN-based and T-based baseline frameworks as well as the experimental setup of these models. Finally, section 3.3 explained the proposed comprehensive evaluation methodology, including the evaluation dimensions, automatic evaluation metrics which are used to evaluate the outputs of the TST systems and a brief description of the human evaluation methodologies.

In the next chapter, we firstly introduce different extensions of the RNN-based baseline model and then conduct some experiments to examine the latent space of the TST frameworks.

These experiments aim at exploring whether or not encoders of TST models encode stylistic information of the input text in their corresponding latent vector. They focus on style domain of sentiment by applying the *Yelp-large* corpus.

# Chapter 4

# The entanglement of sentiment and content

This chapter focuses on the latent space of the adversarial RNN-based TST frameworks in order to explore the separation of style and content within the sentiment domain. The latent space of these sequence-to-sequence encoder-decoder networks can be affected by both encoder and decoder components. Therefore, we propose an encoder variant and a decoder variant of the RNN-based baseline TST architecture (described in section 3.2.1) and compare the baseline model with these two variants throughout the experiments of this chapter. We first look into how these models perform in dealing with the TST task doing a comprehensive evaluation considering content preservation, style-shift and fluency dimensions. Then, we design a probing experiment to investigate the information encoded in their latent space. Finally, we look into how each of these TST frameworks is affected by reinforcing the input during the generation process, which can help us further explore the input latent space. We focus on the sentiment style domain and use the *Yelp-large* corpus introduced in section 3.1.1 throughout these experiments.

# 4.1 Proposed models: an encoder and a decoder variant of the RNN-based TST baseline model

We first introduce two extensions of the RNN-based baseline TST model (introduced in section 3.2.1): a variant of the baseline encoder based on ELMo representations (section 4.1.1), and a style-specific decoder model which applies style-specific decoders instead of a single decoder (section 4.1.2). Then, we evaluate the performance of these two frameworks in section 4.2. Some style-shifted samples created by these TST models are provided in table 4.5.

## 4.1.1 ELMo-based encoder TST model

This research focuses on exploring the input information encoded in the latent space of adversarial encoder-decoder RNN-based architectures when they are applied to the TST problem. The encoder components of these networks create latent representations of given inputs. Therefore, it can be interesting to modify the encoder component of the RNN-based baseline TST model and study how this change affects the latent space of the model.

To do so, we propose an adversarial ELMo-based TST model by extending the baseline model where the encoder is removed and the latent representations of the input sequences are created using a pre-trained ELMo model (Peters et al., 2018). ELMo embeddings are contextualized word representations where the word embeddings are created by combining all the layers of a deep pre-trained bidirectional language model (biLM) (Peters et al., 2018). We employed ELMo embeddings to replace the style task-specific encoder in creating the input vectors, since they have previously been employed in many NLP tasks in the recent years and have achieved promising results.

To create the latent vector for a given input, first, all the tokens of the sequence are mapped to their ELMo representations. Then, the min-mean-max pooling method which

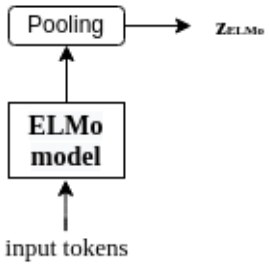


Figure 4.1 ELMo-based embeddding for each input as the vector to initialize the decoder of the ELMo-based encoder TST model

is described in section 3.3.1.3 is employed to combine these token embeddings into the sequence latent vector $\mathbf{z_{ELMO}}$ (figure 4.1). The generator block of the ELMo-based model contains an RNN which acts as the decoder component and is initialised by $\mathbf{z_{ELMO}}$ and the desired style of the output text. The adversarial block of this framework is the same as that of the baseline model. While training, the backpropagation is done by using equation 3.6 except for the fact that only the parameters $\theta_D$ are estimated, since we do not train an encoder in this model. The two style-specific discriminators are trained jointly with the generator block (the same as in the baseline model). To update the parameters of $\theta_{Disc_S}$ ($s \in \{s_1, s_2\}$), equation 3.2 for $\mathbf{Disc_{s_1}}$ and its symmetric equation for $\mathbf{Disc_{s_2}}$ are used.

### 4.1.2 Style-specific decoders TST model

Encoder and decoder can both affect the latent space of an encoder-decoder sequence-to-sequence architecture. To further investigate their effect, we modify the decoder of the RNN-based baseline TST model in this section and study how this change affects the latent space of the model. To do so, we extend the baseline model to a multi-decoder framework where a separate decoder is employed in the generator block for each style[1]. The generator block of the proposed model, therefore, consists of *(i)* two style-specific decoders $\mathbf{D_{s1}}$ and $\mathbf{D_{s2}}$ and *(ii)* the encoder which is shared between $\mathbf{D_{s1}}$, and $\mathbf{D_{s2}}$ as depicted in figure 4.2.

[1]The code and data are available at https://github.com/somayeJ/RNN-based-TST-experiments

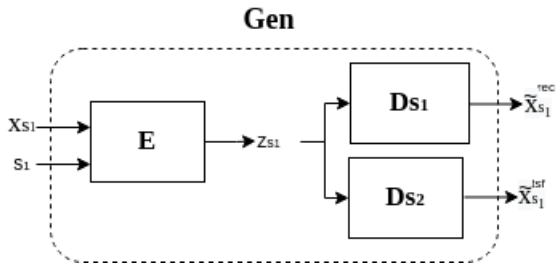


Figure 4.2 The schema of the *Gen* (generator block) of the multi-decoder RNN-based TST model.

The discriminator block is the same as that of the base model and contains style-specific classifiers $\mathbf{Disc_{s1}}$, and $\mathbf{Disc_{s2}}$.

While training, for each given input $\mathbf{x_{s1}}$ with the source style $s_1$, $\mathbf{D_{s1}}$, and $\mathbf{D_{s2}}$ are initialized with $\mathbf{z_{s1}}$ as the latent representation of the input created by the encoder subnetwork. $\mathbf{D_{s1}}$ generates an output sequence in the style $s_1$, i.e. it reconstructs the input sequence and $\mathbf{D_{s2}}$ creates a style-shifted sequence in the style $s_2$. Generation of the outputs for a given sequence $\mathbf{x_{s2}}$ with the source style $s_2$ is done in the same manner where each style-specific decoder creates an output in its specific style.

The reconstruction loss is computed for this model as the summation of the two reconstruction losses corresponding to each decoder, equation 4.1 shows the reconstruction loss $L_{rec_{s_1}}$ computed for $\mathbf{D_{s1}}$ ($L_{rec_{s_2}}$ is computed symmetrically).

$$\mathscr{L}_{rec_{s_1}} = \quad -\log \Pr_{D_{s_1}}(\tilde{\mathbf{x}}_{s_1}^{(rec)} | x_{s_1}) \tag{4.1}$$

The joint training of the discriminator and generator block is done in a similar fashion to that of the base model by using the equations 4.2 which computes the total loss as the summation of the adversarial losses (equations 3.4 and 3.5) and reconstruction losses to update the parameters of the $(\theta_E, \theta_{D_{s1}}, \theta_{D_{s2}})$. Equation 3.2 and its symmetrical equation are used to estimate $(\theta_{Discs_1}, \theta_{Discs_2})$ of the style-specific discriminators.

$$\mathscr{L}_{total}(\theta_E, \theta_{D_{s1}}, \theta_{D_{s2}}) = \quad \mathscr{L}_{rec_{s_1}} + \mathscr{L}_{rec_{s_2}} + \mathscr{L}_{adv,s_1} + \mathscr{L}_{adv,s_2} \tag{4.2}$$

**Experimental setup of the proposed models** The hidden state size of the uni-directional GRU cells of the encoder and decoders of the *multi-decoder model* (section 4.1.2) is 700. In the *ELMo-based model* (section 4.1.1), the hidden state size of the uni-directional GRU cells is set to 3072. The other parameter settings of these two models are similar to those of the RNN-based baseline model which are reported in the table 3.8.

## 4.2 Evaluating the proposed frameworks

This section first reports the results of the baseline, ELMo-based encoder and multi-decoder systems. Then it investigates the validity of the evaluation methodology by conducting human evaluation tests. To do the evaluation, the restaurant review dataset *Yelp-large* was used (described in section 3.1.1).

### 4.2.1 Automatic evaluation

To investigate the performance of the TST models automatically, we considered the three evaluation aspects of fluency, style-shift strength *SSP*, and content preservation power *CPP*. We computed how well the style-shifted sequences preserve the content of the input sequences by using the GloVe-based *CPP* metric. We also measured how each model shifts the style of the inputs to a given desired style by computing the accuracy of the pre-trained classifier in labelling the style-shifted sequences with this desired style. Finally, the fluency of the style-shifted sequences are measured by calculating the perplexities that a pre-trained language model assigns to them where lower values represent more fluent sequences (more details in section 3.3).

| | Model | SSP | PPLX | CPP (GloVe-based) |
|---|---|---|---|---|
| (a) | Base model | 78.8% | 43.5 | 0.925 |
| (b) | ELMo-based encoder | 39.3% | 134.5 | 0.895 |
| (c) | Multi-decoder | **94.4%** | **39.1** | 0.910 |

Table 4.1 The results of automatic evaluation considering three aspects of *SSP*, *CPP* and fluency (PPLX). Higher values of *SSP* and *CPP* signify better performance of the models, whereas, for fluency, lower scores are better.

Table 4.1 represents the results of the baseline, ELMo-based encoder, and Multi-decoder TST models in rows *a* to *c*.These results show that the ELMo-based encoder model performs worse than both the base model and multi-decoder framework across the three aspects of evaluation. This indicates that, although ELMo embeddings have improved many NLP-related tasks in recent years, TST is reliant on the task-specific encoder subnetworks to create the embeddings of the input sequences.

The multi-decoder framework outperforms the base model in terms of shifting the style and fluency but works slightly worse considering *CPP*. This is mainly because each style-specific decoder learns the distribution of a specific-style data and generates in that style.

To better interpret the scores of *CPP*, we can compare the lower bound (*LB*) of the GloVe-based *CPP* score of the Yelp-*large* dataset (0.84) with the GloVe-based *CPP* scores of the baseline model and its extensions. This shows that all the models have a better performance in preserving the content than random.

### 4.2.2 Human evaluation

We conducted three human evaluation tests corresponding to the three evaluation dimensions. These tests were done in a totally blind manner, i.e. samples and models were shuffled for each test. This was to ensure that the evaluators would not be biased while doing the test. Due to ELMo's bad performance and the time constraints of the human evaluation, we did not include the results of the ELMo model in this test. The total number of samples considered was 450: 150 samples (75 samples from each style) randomly selected from the

Yelp test set, as well as, their corresponding 300 style-shifted sequences generated by the base and multi-decoder models. The scores reported in each test are computed by taking the average over the labels of the judges over all the samples. For each of the three human tests, the Krippendorff's inter-rater agreement (Krippendorff, 1980) is computed which gives an insight on the level of ambiguity of the test and therefore the level of its validity.

**Style-shift power (SSP)** In this test the 29 participants were provided with one style-shifted sample at a time and were asked the following question:

- *Question:* What is the sentiment of this sequence?
- *The possible labels*: "positive", "negative", or "neutral".

The judges labelled the sequences generated by the base model as having the desired style in 58.3% of the cases. This number raised to 67.6% for the multi-decoder model. The inter-rater agreement of this test was 0.752.

**Fluency** The 25 evaluators in this test were provided with one style-shifted sample at a time and were asked the following question:

- *Question:* How grammatically correct is the given sequence?
- *The possible labels*: "incorrect", "partly correct" and "correct".

The results of this test with the inter-rater agreement of 0.568 shows that the multi-decoder model outperforms the base model in terms of fluency and grammatical correctness. Multi-decoder outputs were labelled by the annotators 67.6% of the times as "correct" as compared to the outputs of the baseline model which were labelled as "correct" 64.2% of the times. The performance of the models is computed after disregarding the "partly correct" label due to its ambiguity for the judges.

| Models | SSP | CPP | Fluency |
|---|---|---|---|
| Baseline model | 58.3% | 52.2% | 64.2% |
| Multi-decoder model | 67.6% | 41.6% | 67.6% |
| Number of raters | 29 | 22 | 25 |
| inter-rater agreement | 0.752 | 0.772 | 0.568 |

Table 4.2 **SSP**: Percentage of the times each system output was labelled with the correct desired style by the judges. **CPP** : Percentage of the times each system output was labelled as having the same content with the input by the judges. **Fluency**: Percentage of the times each system output was labelled as having the correct grammatical structure by the judges.

**Content preservation power (CPP)** The participants of this test were provided with the style-shifted outputs of the base and multi-decoder models and their corresponding input sample from the test set of the *Yelp-large* and were asked the following question:

- *Question:* Which sequence most closely resembles the source sentence in terms of content (disregarding the sentiment)?
- *The possible labels*: “equally good", “equally bad", “first sample is better", or “second sample is better".

The 22 participants of this comparative test had an the inter-rater agreement of 0.772 and labelled the outputs as being more similar to the source sequence in 52.2% of the cases for the base model and 41.6% of cases for the multi-decoder model.

The results of the human test show that the annotators and automatic metrics rank the models similarly. This validates our automatic evaluation methodology (section 3.3).

## 4.3 Investigating the relation between style and content

The main goal of this experiment is to investigate to what extent the latent representations of the inputs generated by the baseline, ELMo and multi-decoder TST models encode stylistic information. We conduct the following experiments in sections 4.3.1 and 4.3.2 to examine the style-content separation within these models.

### 4.3.1 Reinforcing z during generation

In this experiment, we study how reinforcing the latent vector representations of the input at each generation step affects the performance of the base and multi-decoder models in particular in terms of preserving the content and shifting the style of the inputs. Moreover, we aim at shedding more light on the latent space of the TST frameworks, in terms of the relationship between the features that are encoded in the **z** vectors and the performance of the models.

In standard sequence-to-sequence models, the **z** vector is only used to initialise the decoder. This means that, in the first generation step, as the input, the GRU cell of the base model takes the `<start>` token, the **z** and the target style vectors, while the GRU cell of the multi-decoder model only takes the `<start>` token and the **z** vector. Both models output the first token of the style-shifted sequences and use this generated output as the input for the next generation step. Consequently, the input information received by **z** tends to fade from the evolving hidden state used by the decoder because at each generation step the only input is the output of the previous step. Here, we introduce reinforced variants of the baseline and multi-decoder models where the **z** vector, as illustrated in figure 4.3, is re-inputted to the decoder while generating each output token. To do so, we follow the merging strategy proposed by Tanti et al. (2018) and, at each generation step, we concatenate the latent representation of the input sequences **z** to the logit vectors of the decoder GRU cells before feeding them to the projection layer.

Extending the baseline model and multi-decoder framework to reinforce the input content during each generation step makes the features of the **z** vectors more present in the reinforced model which intensifies the presence of the input information while doing the generation. Comparing rows *d* and *e* to rows *a* and *c* of the table 4.3 shows how this modification affects the performance of the baseline and multi-decoder models. Moreover, comparing rows *d* and

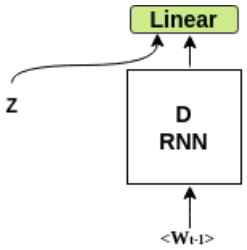


Figure 4.3 Reinforcing $z$ while generating tokens in each step of generation, as an instance step $t$

| | Model | SSP | PPLX | CPP (GloVe-based) |
|---|---|---|---|---|
| (a) | Base model | 78.8% | 43.5 | 0.925 |
| (b) | ELMo-based encoder | 39.3% | 134.5 | 0.895 |
| (c) | Multi-decoder | **94.4%** | **39.1** | 0.910 |
| (d) | Base model (reinforced) | 66.0% | 49.6 | **0.930** |
| (e) | Multi-decoder (reinforced) | 90.7% | 83.0 | 0.860 |

Table 4.3 Comparing the performance of the base and multi-decoder models to their reinforced versions where the input vector $\mathbf{z}$ is injected to the decoder(s) at each step of generation.

*e* to rows *a* and *c* of the table 4.4 show how emphasizing the input vector during generation modifies the presence of the source style in the latent space of these two TST systems.

### 4.3.2 Probing the disentanglement of style and content

We follow the same strategy employed by Conneau et al. (2018) to conduct a probing classification experiment in order to analyse what the baseline model, ELMo-based encoder system and multi-decoder model encode in their latent space. We train a separate classifier for each of the three TST networks. Each classifier is a Feed-Forward network with a single hidden layer and a sigmoid output layer and is trained to infer the source style $s$ from a latent vector $\mathbf{z_s}$ generated by the encoder of its corresponding TST model for the input text $x_s$ (figure 4.4). The classifier corresponding to each model is trained using the training and development sets of the *Yelp-large* corpus. The probing classification scores are then computed as the accuracy of their classifiers on detecting the source labels on the test set of

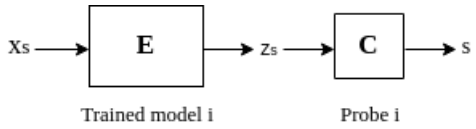


Figure 4.4 Probing classifier *i* which is trained to learn the source style of the latent vectors of the input created by *E* of the TST model *i*.

| | Model | Accuracy |
|---|---|---|
| (a) | Base model | **99.97**% |
| (b) | ELMo-based encoder | 93.84% |
| (c) | Multi-decoder | 97.56% |
| (d) | Base model (reinforce) | 98.39% |
| (e) | Multi-decoder (reinforce) | 91.85% |

Table 4.4 The accuracy of probing classifiers corresponding to each TST framework in detecting the source style in the latent space of the TST models.

the *Yelp-large* corpus. If no stylistic information is present in the latent vectors of a given model's encoder, its probing classifier is expected to have a low accuracy.

Rows *a*, *b*, and *c* of table 4.4 show the accuracy of the classifiers trained for the baseline, ELMo-based encoder and multi-decoder models. The higher accuracy of the probing classifier corresponding to the baseline model as compared with the classifiers corresponding to the multi-decoder and ELMo-based encoder systems indicates the presence of more source stylistic features in the latent space of this model. Similarly, the accuracy of the ELMo-based encoder classifier implies that the least input stylistics features are present in the input latent representations of this model.

## 4.4 Discussion

The main focus of this chapter is exploring the latent space of adversarial end-to-end encoder-decoder RNN-based TST systems and investigating the separation of content and sentiment in their input latent vectors. Modifying encoder and decoder components can affect the input latent space of these TST models and studying these effects can shed more light on the

latent space and what is encoded in it. Therefore, the ELMo-based encoder model, and the multi-decoder framework were introduced. To explore the latent space of the TST systems, we conducted two main experiments while taking sentiment as the case study. The first experiment was a probing classification task which investigates the presence of the source style in the input latent space and the second experiment focused on reinforcing the input representation **z** throughout the generation process while creating each style-shifted token. The results from these experiments led to the following conclusions.

- The drop of the *SSP* across the baseline system and multi-decoder model when the input representation **z** is reinforced during generation (rows *d* and *e* of table 4.3) indicates that latent representations of the inputs are not free of the source style. Therefore, due to the presence of both style and content features in the latent space, **z**-reinforcement negatively affects the *SSP* and may improve the *CPP* as in the case of the baseline model (row *d* of table 4.1).

- Rows *d* and *e* of table 4.3 show the fluency drops across both of the models when reinforcing is applied. This indicates that **z**-reinforcement inhibits the fluency of the decoder(s). This implies that **z** contains features from a language distribution which differs from the language distribution to which the style-shifted sequences belong, i.e. source as opposed to the target style language distributions and the presence of the source style features in the **z** vectors confuses the decoder language models.

- Table 4.4 reports high accuracy for the probing classifiers corresponding to all the TST models. This indicates the presence of the source style in their latent space implying that style and content are entangled.

- Rows *d* and *e* of table 4.4 illustrates that the accuracy of the classifiers corresponding to the reinforced-based and reinforced multi-decoder models in labelling source style lowers compared to the base and multi-decoder systems (rows *a* and *c* of table 4.4).

This firstly indicate that the reinforced models encode less source stylistic features in their latent vectors. We hypothesize that this happens since the repetitive reinforcement of the source style encoded in **z** vectors throughout the generation process interferes with the ability of the model to shift the style due to the presence of the source style in the **z** space. Therefore, the model learns to strip out the source style from the **z** space during training in order to reduce the confusion these source style features cause when **z** is reinforced during generation. Also, we notice that encoding appreciably less source style in the **z** of reinforced multi-decoder model leads to a drop in the *CPP* of this model (table 4.3 row *e*) which provides another proof that style and content are not totally separable.

These observations imply the presence of an entanglement between the style and content, i.e. source style is not totally separable from the content. These results provide us with a good insight about the main research focus of this chapter which also addresses the first research question of this thesis (section 1.2). The other findings of this chapter are discussed in the following.

- The results in table 4.4 showed how the **z** vectors generated by the task-specific encoders differ from the ELMo-based **z** vectors. Higher accuracy scores of the classifiers trained with the **z** vectors generated by the baseline model (with and without reinforcing) and multi-decoder model (with and without reinforcing) show that task-specific encoders encode more source stylistic features within the input embedding representations compared to the ELMo-based encoder model. This suggests that the role of the encoders in the TST problem is not only to encode the content-related information, but also, to mark the source style features which guides the TST system while generating in the desired style.

- Table 4.4 illustrates lower accuracy for the classifiers which correspond to the multi-decoder models compared to their corresponding single-decoder model (rows *c* and *e*

compared to the rows *a* and *d*, respectively). This indicates that style-specific decoder frameworks need less of the source style indicators in their **z** space compared to the single-decoder frameworks, since their decoder learn to generate in a specific style.

- The results in table 4.4 indicate that the reinforcing of **z** results in both the base and multi-decoder frameworks learning to reduce the amount of information about the input style encoded in the **z** representation (a reduction of approximately 1.6% probing accuracy for the base model and a reduction of approximately 5.7% probing accuracy for the multi-decoder framework). Focusing now on table 4.1, these reductions in the encoding of the input style in **z** correspond with a decrease in *SSP* (reinforcing in the base model results in a drop of 12.8% in *SSP* and in the multi-decoder model reinforcing **z** during generation results in a drop of 3.7% in *SSP*). We hypothesize that the relatively small reduction in the probing accuracy accompanied by the relatively large reduction in *SSP* observed for the base model is due to the fact that, as noted above, single-decoder models are more reliant on the source stylistic features while doing the task compared to the models with multiple decoders (i.e. because the base model only have a single decoder that generates in both styles, the presence of input style in **z** can be beneficial as a signal to the decoder to generate in the other style).

- Rows *a* and *c* of the tables 4.1 and 4.4 show that the higher the accuracy of a probing classifier the better its corresponding model performs on preserving content. This indicates a direct relation between the amount of the source style encoded in the **z** vectors of the base and multi-decoder system and their *CPP* which again implies an entanglement between source style and input content. This is further validated by the results of the **z**-reinforcement experiment. We investigate this observation more in chapter 5 by studying the relation across other TST frameworks and style domains.

- The findings of this chapter (table 4.3) also imply a trade-off between evaluation dimensions where an increase in *CPP* can lead to a drop in *SSP* and also worse fluency. This highlights the importance of using a comprehensive evaluation methodology for TST problem similar and further validates the methodology proposed in this chapter. This also motivates the use of this evaluation methodology throughout the experiments of the later chapters.

- We conducted a human evaluation test considering the three evaluation aspects. The results were inline with the results of the automatic evaluation which confirmed the validity of our automatic evaluation methodology. We use this validated automatic evaluation methodology to investigate the performance of TST frameworks throughout the experiments of this chapter and later chapters.

- The results of human evaluation are inline with the intuition that style and content are entangled. To be more precise, on one hand, rows 1 and 2 of table 4.2 show that according to the human judges the base model performs better in preserving the content as compared to the multi-encoder system. The results of the probing experiments, on the other hand, report a higher probing accuracy score for the baseline model (rows *a* of table 4.4) as compared to the multi-encoder system (rows *c* of table 4.4). This suggests that the presence of more source style in the latent space of the baseline model leads to better *CPP* which further confirms the style-content entanglement.

## 4.5 Conclusion

Studying the latent space of the generation block of RNN-based frameworks, as the main focus of this chapter, indicated that for the TST task, style and content are entangled elements and their total separation is not possible while considering sentiment as the style domain. This observation shifted our attention towards examining whether these findings hold across

other style domains and architectures. In chapter 5, we extend our experiments to investigate the entanglement of style and content in the style domain of formality while implementing other variations of RNN-based TST frameworks.

| **From negative to positive** |
|---|
| 1. so nasty . |
| 2. so awesome . |
| 3. ugh delicious. |
| 4. so delicious . |
| 5. so fun . |
| 6. so awesome ! |
| |
| 1. my goodness it was so gross . |
| 2. my husband was so delicious . |
| 3. my experience was great . |
| 4. my goodness it was amazing . |
| 5. my goodness was amazing . |
| 6. my , it was also |
| |
| 1. the cake portion was extremely light and a bit dry . |
| 2. the prime rib was very smooth and very reasonable . |
| 3. the pizza is fresh and a very nice . |
| 4. the tuna was a bit dry and satisfying . |
| 5. the portion was extremely tasty and reasonably priced . |
| 6. the crust was very dry and dry a bit . |
| **From positive to negative** |
| 1. i highly recommend this place ! |
| 2. i wo n't be it . |
| 3. i do n't even go back . |
| 4. i do not recommend this place ! |
| 5. i highly recommend this place place ! |
| 6. i loved not recommend this place . |
| |
| 1. my appetizer was also very good and unique . |
| 2. my chicken was just a little hot and texture . |
| 3. my pie is just thin and just like pie . |
| 4. my boyfriend was n't and had a very dry . |
| 5. my entree was very unique and also very good . |
| 6. my appetizer was also very good and lacked lacked lacked lacked . |
| |
| 1. the food is fresh and the environment is good . |
| 2. the food was tasty and the quality is pretty expensive . |
| 3.the food was rude and do n't waste the time . |
| 4. the food is fresh and the sandwich was too salty . |
| 5. the food is good and the food is not good . |
| 6. the food is the food and the food is the food comes . |

Table 4.5 Style-shifted outputs of the TST models using *Yelp-large*. 1: input sequence, 2: Multi-decoder model, 3: Reinforced multi-decoder model 4: Baseline model 5: Reinforced baseline model and 6: ELMo-based encoder model.

# Chapter 5

# Investigating the style transfer task in sentiment domain versus formality domain

This chapter focuses on exploring whether the findings of chapter 4 can be extended to other style domains, i.e. it investigates whether style is a consistent concept across various domains. To do this, we do a series of experiments which mainly focus on exploring the latent space of sentiment-transfer and formality-transfer models. As the first step, we modify the encoder of RNN-based baseline model (described in section 3.2.1) and propose a multi-encoder system and an attention-based framework. The reason being that the input latent representations which are the focus of our experiments throughout this chapter can be highly affected by the encoder architecture and studying these two systems enables us to examine the effect of these modifications.

The experiments of this chapetr aim at exploring the style domain of sentiment as well as the style domain of formality, specifically in terms of how sentiment and content are entangled compared to formality and content. They involve investigating whether the previously observed sentiment-content entanglement holds across the multi-encoder and attention-based

TST frameworks. They also examine the presence of the source stylistic features in the latent space of the TST models and their relation with the content across sentiment and formality domains. This chapter also explores sentiment-shift versus formality-shift tasks and further validate the comprehensive evaluation methodology (described in section 3.3) by employing different techniques to confirm the validity of *CPP* metrics. Throughout the experiments of this chapter, we use the *Yelp-large* corpus (described in section 3.1.1) and *GYAFC*-$v_1$ and *GYAFC*-$v_2$ corpora (described in section 3.1.2).

## 5.1 Implementing more powerful encoders

Encoders of encoder-decoder sequence-to-sequence frameworks create input latent representations. So, the architecture of these networks can affect these latent vectors. In this section, we extend the baseline model (described in section 3.2.1) and implement two encoder variants: a multi-encoder framework (section 5.1.1), and an attention-based model (section 5.1.2). Employing these TST models throughout the experiments of this chapter enables us to study the effect of this modification on the latent space, which is the main focus of this chapter.

### 5.1.1 Multi-encoder framework

We extend the baseline model by employing one RNN as the encoder for each style in the system. Hence, the generator block **Gen** of the proposed model contains two style-specific encoders $\mathbf{E_{s1}}$ and $\mathbf{E_{s2}}$, as well as a decoder **D** which is an RNN shared between the two encoders. The discriminator block **Disc** of the proposed framework is the same as that of the baseline model and functions similarly. It consists of style-specific discriminators $\mathbf{Disc_s}$ ($s \in \{s_1, s_2\}$).

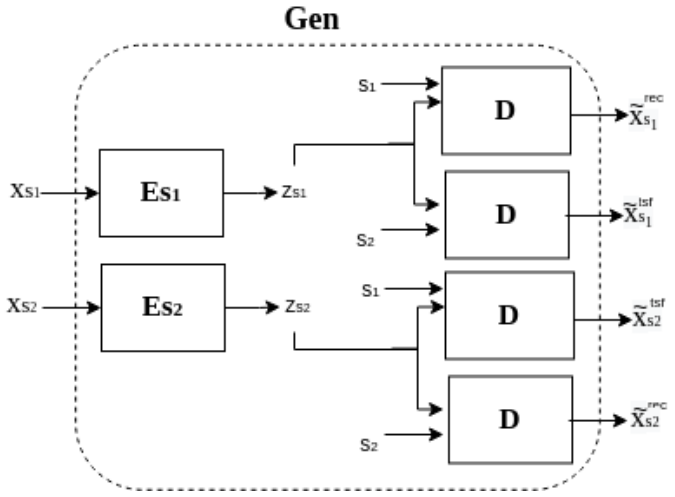


Figure 5.1 The schema of the *Gen* (generator block) of the multi-encoder RNN-based TST model.

As figure 5.1 depicts, while training, an input pair $(\mathbf{x}_{s_1}, \mathbf{x}_{s_2})$ is encoded by the style-specific encoders $(\mathbf{z}_{s_1}, \mathbf{z}_{s_2}) = (\mathbf{E}(\mathbf{x_{s_1}}), \mathbf{E}(\mathbf{x_{s_2}}))$. Then, the reconstruction and style-shifted sequences of each input are generated by conditioning the decoder on $\mathbf{z_{s1}}$ and $\mathbf{s_1}$ or $\mathbf{z_{s1}}$ and $\mathbf{s_2}$, respectively (similarly for $\mathbf{x_{s_2}}$). The reconstruction loss is the summation of the $L_{rec1}$ (equation 5.1) and $L_{rec2}$ (symmetrically computed).

$$\mathscr{L}_{rec1}(\theta_{E_{s1}}, \theta_D) = \quad -\log \Pr(\widetilde{\mathbf{x}_{\mathbf{s_1}}}^{(rec)} | \mathbf{x_{s_1}}) \tag{5.1}$$

Adversarial training of this system is done in a similar fashion to the baseline model. Also, similar to the baseline model, **Disc** and **Gen** are jointly trained by updating the parameters of $\theta_{Disc_{s1}}$ and $\theta_{Disc_{s2}}$ using the equation 3.2 ($s \in \{s_1, s_2\}$) and estimating the parameters $\theta_{E_{s1}}$, $\theta_{E_{s2}}$ and $\theta_D$ using the following equation 5.2 where the adversarial loss similar to the baseline model is computed by equations 3.4 and 3.5 and is applied to guide the training process into creating the style-shifted sequences.

$$\mathscr{L}_{total}(\theta_{E_1}, \theta_{E_2}, \theta_D) = \quad \mathscr{L}_{rec_{s1}} + \mathscr{L}_{rec_{s2}} + \mathscr{L}_{adv,s_1} + \mathscr{L}_{adv,s_2} \tag{5.2}$$

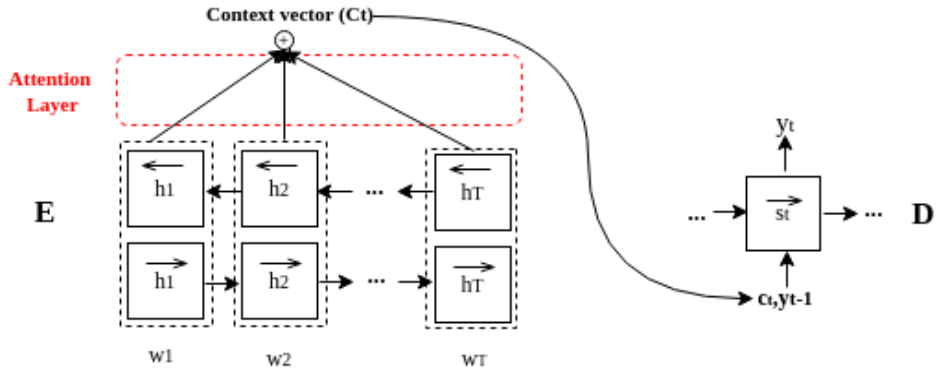


Figure 5.2 Generating the output token at time step $t$ while creating $c_t$ considering $s_{t-1}$

### 5.1.2 Attention-based model

We also extend the baseline model in another way by implementing the attention strategy[1] proposed by Bahdanau et al. (2015) and introduce the attention-based model which consists of an encoder **E**, a decoder **D** and style-specific discriminators $\mathbf{Disc_s}$ ($s \in \{s_1, s_2\}$).

**Encoder (E):** **E** is a single-layered bi-directional subnetwork that contains two RNNs which read an input sequence $\mathbf{x}_s$ ($s \in \{s_1, s_2\}$) in both forward and backward directions and are initialized by the dense vector of the source style $s$. If the length of $\mathbf{x}_s$ is $T$, the encoder output states $h_1, \ldots, h_T$ are formed by the concatenation of the outputs from forward and backward cells, $h_i = (\overrightarrow{h_i} \oplus \overleftarrow{h_i})$. The latent representation denoted as $\mathbf{z_s}$ is created by the concatenation of the last state of the two RNNs of **E**, $z_s = (\overrightarrow{z_s} \oplus \overleftarrow{z_s})$. Theses vectors are then augmented as $h'_1, h'_2, \ldots, h'_T$ and $\mathbf{z'_s}$ by concatenating the $h_1, \ldots, h_T$ and $\mathbf{z_s}$ with the dense vector of the desired style.

**Decoder (D):** **D** is a single-layered uni-directional RNN which takes the dense vector of the desired style $s$ as the initial state and at each time step, for instance the time step $i$ (figure 5.2), considering the previous state $s_{i-1}$, it creates the context vector $c_i$ as latent representations of the input sequence by doing the weighted summation of $h'_1, h'_2, \ldots, h'_T$

[1]The attention mechanism is described in details in section 2.1.4.2

where at time step 0, $\mathbf{z}'_{\mathbf{s}}$ is considered as the previous state. The weights assigned to the augmented state $h'_j$ of **E**, at time step $i$ is calculated by equation 5.3 where the $score_{i,j}$ is normalized with regards to the scores assigned to each augmented state $h'_1, h'_2, \ldots, h'_T$ of the input using a softmax layer.

$$a_{i,j} = \frac{\exp(score_{i,j})}{\sum_{k=1}^{T} \exp(score_{i,k})} \tag{5.3}$$

The score for $h'_j$ at time step $i$ ($score_{i,j}$) is computed relative to the content of the previous **D** state ($s_{i-1}$) by passing the $h'_j$ and $s_{i-1}$ through a two-layer feed forward network where the activation function of the first layer is a hyperbolic tangent.

$$score_{i,j} = W_f(\tanh((W_s s_{i-1} + b_s) + (W_h h'_j + b_h))) \tag{5.4}$$

The discriminator block (**Disc**) of the attention-based model is similar to that of the base system in terms of the architecture, functionality and training regime. The training process is similar to the training regime of the base model. During the training steps of this model the parameters and weights of the attentions layer which consists of fully connected Feed-Forward layers are jointly trained with other components of the **Gen** and updated using the equation 5.4.

**Experimental setup of the proposed models** The hidden state size of the uni-directional GRU cells of the encoder and decoders are set to 700 in the *multi-encoder model* (section 5.1.1). In the attention-based model (section 5.1.2) the size of both bi-directional GRU cells of the encoder and uni-directional cells of the decoder are also set to 700. The other parameter settings of these two models are similar to those of the RNN-based baseline model which is reported in table 3.8.

## 5.2 Evaluating the proposed frameworks

In this section, we compare the sentiment-shift and formality-shift tasks by studying the performance of the baseline model and its two proposed extensions. We use the *Yelp-large* (section 3.1.1) and *GYAFC-$v_2$* (section 3.1.2) corpora to evaluate how these models perform across sentiment and formality domains by applying the evaluation methodology (described in section 3.3). In section 4.2.2, we showed that the comprehensive evaluation methodology is in line with human judgement. Here, we would like to further confirm this methodology by using different techniques to compute and validate *CPP*[2].

The GloVe-based *CPP* metric has been widely used in the previous TST researches including (Fu et al., 2018a; Shen et al., 2017; John et al., 2019; Jafaritazehjani et al., 2020, 2021, 2022). However, while using this metric, the variance of scores is small, interval of the $(0.84, 1)$ (considering the lower and upper bounds of the metric). This leads to a lack of sensitivity which questions the efficiency and precision of this metric (John et al., 2019). To validate the GloVe-based *CPP* scores, we previously showed (section 4.2.2) that GloVe-based *CPP* scores rank the models similar to human annotators. Here, we employ two other *CPP* metrics, *WMD* and *WO*, which apply different techniques and investigate how these metrics rank the TST models compared to GloVe-based metric. We also use confidence intervals to assess whether differences in GloVe-based *CPP* scores between different architectures are statistically different.

As described in details in section 3.3.1.3, *WO* is a unigram based metric which is computed after filtering the stop words from the given input and its corresponding style-shifted text. To compute *WMD* which is a special case of Earth Mover's Distance (Rubner et al., 2000), we follow the approach explained in section 3.3.1.3, by considering inputs and their corresponding style-shifted text. The GloVe-based *CPP* score is also computed as explained in section 3.3.1.3 by computing a cosine similarity score between the embedding

[2]Some style-shifted samples created by the TST models discussed in this chapter are provided in tables 5.5 and 5.6.

| **Datasets** | | ***Yelp-large*** | | | ***GYAFC-$v_2$*** | | |
|---|---|---|---|---|---|---|---|
| Models | | *Baseline* | *Multi-E* | *Att-based* | *Baseline* | *Multi-E* | *Att-based* |
| **SSP** | | **78.76%** | 76.26% | 53.99% | **65.11%** | 58.82% | 59.58% |
| **PPL** | | **44.52** | 46.91 | 48.11 | 26.81 | 28.84 | **23.72** |
| **CPP** | WMD | 0.695 | 0.647 | **0.3827** | 0.7783 | **0.7733** | 0.9098 |
| | WMD-LB | 1.161 | | | 1.1224 | | |
| | WO | 0.199 | 0.254 | **0.475** | 0.0645 | **0.0675** | 0.0288 |
| | WO-LB | 0.01 | | | 0.0045 | | |
| | GloVe | 0.9239 | 0.9311 | **0.9542** | 0.9088 | **0.911** | 0.8869 |
| | GloVe-LB | 0.84 | | | 0.8792 | | |

Table 5.1 Evaluation results of the baseline model, *Att-based* (attention-based) and *Multi-E* (multi-encoder) models. Higher values in the table show better performance except for the metric *WMD* and *PPL*. LB indicates the Lower Bound of different *CPP* metrics.

representations of each pair of input and style-shifted output. Table 5.1 lists the results for the baseline, multi-decoder and attention-based models on the *Yelp-large* and *GYAFC-$v_2$* datasets. In terms of *CPP* metrics, one observation that can be taken from these results is that the *WMD*, *WO* and GloVe-based metrics are in agreement in terms of rank order of the systems. On the *Yelp-large* dataset, all these *CPP* metrics rank the attention-based model best, followed by the multi-encoder and then the baseline model. On *GYAFC-$v_2$* dataset, all theses metrics rank the multi-encoder model best, then the baseline model and finally the attention-based model. This consistency across these three different *CPP* metrics validates the use of GloVe-based *CPP*.

We take a further step and examine whether the differences reported for the GloVe-based *CPP* scores for the TST models across one style domain are statistically significant. To do so, for each style domain and TST framework, we compute the confidence intervals (CI) for the GloVe-based *CPP* scores. CIs provides a range of estimates for the true mean of a population, centred on the sample mean, and is defined as an interval with a lower bound and an upper bound. The interval is computed at a designated confidence level. The confidence level represents the long-run frequency of confidence intervals that contain the true value of the parameter. In other words, 99% of CIs computed at the 99% confidence level contain the

true population mean. Given a sample mean value $m$, the sample standard deviation $\sigma$ and the sample size $n$, the confidence interval is defined by the following equation:

$$Cl = m \pm Z\frac{\sigma}{\sqrt{n}} \tag{5.5}$$

In this equation $Z$ is the critical value, which depends on the desired confidence level, for instance, for a 99% confidence level it is 2.576, as provided by a $Z$ table. $Z$ tables differ on usage, but essentially, the table tells us what the critical value is for many common probabilities. Note that the factors affecting the width of the CI include the confidence level, the sample size, and the variability in the sample. Larger samples produce narrower confidence intervals when all other factors are equal. Greater variability in the sample produces wider confidence intervals when all other factors are equal. A higher confidence level produces wider confidence intervals when all other factors are equal. Thus, calculating the CI for a single mean will provide a range within which the true mean can be found[3].

To compute the CIs for the GloVe-based *CPP* scores, first, the GloVe-based *CPP* scores of each of the pairs of input, and style-shifted output sequences are computed across the test set of the considered domain of style. Then, the CIs around the average model performance is measured. We observed that for the *Yelp-large* dataset, across the three frameworks the CIs computed for GloVe-based *CPP* scores do not overlap with the confidence level of 0.99. This confirms the validity of the variations between the *CPP* scores of the baseline, multi-encoder and attention-based models while using the *Yelp-large* data. The confidence intervals do not have an overlap for *GYAFC-*$v_2$ with the confidence level of 0.8.

For each of the *CPP* metrics and for each dataset, we calculate a lower bound (*LB*) score using the method explained in section 3.3.1.4. *LB* scores provide us with a better insight of how TST models perform in preserving the content of the input text while shifting its style.

[3]An example in the context of confidence intervals can be found here: https://www.mathsisfun.com/data/confidence-interval.html

Overall, the *CPP* scores of all frameworks are higher than *LB* scores. However, the *CPP* scores computed for the attention-based model in the domain of formality is very close to the *LB* scores considering the three metrics of $WO$, $WMD$ and GloVe-based, i.e. the *CPP* scores of attention-based model is closer to *LB* scores rather than the the *CPP* scores of the other two TST models in the domain of formality. This can mean that the model has not reached convergence, i.e. it needs either a longer training time or more data. To keep the training parameters constant; such as number of epochs, we reported the results of this model as is without trying different techniques to reach better performance.

Excluding the attention-based TST model, the results listed in table 5.1 indicate the following. First, we observe that an increase in *CPP* results in a drop in *SSP*, i.e. in both domains of formality and sentiment, *CPP* and *SSP* appear to be inversely related. Also, the results of table 5.1 indicate that in both style domains, fluency of the style-shifted text generated by each model gets worse as the *CPP* of the model improves, i.e. better values of *CPP* leads to higher perplexity scores. This is in line with the trade-off observed in the results of the previous chapter (section 4.5). The observation that the trade-off holds while applying other architectures as well as across other style domains and not only sentiment reinforces the necessity of applying a comprehensive evaluation methodology, i.e. taking the three evaluation dimensions into account for the TST problem.

Finally, the results in table 5.1 suggest that applying style-specific encoders leads to higher *CPP* scores in the sentiment and formality domains. Moreover, comparing the results of the baseline model and its two extensions highlights the performance variation of the attention-based architecture versus multi-encoder across these two style domains. This means that employing the attention-based technique affects the sentiment-shift and formality-shift tasks differently as compared with using the multi-encoder strategy. To be more precise, in the sentiment domain, employing the attention-based technique has a larger influence on the performance of the model versus employing multi-encoder architecture, i.e. we

observe a larger increase in *CPP* values and a bigger drop in *SSP* score while employing attention-based as compared to the multi-encoder system. This difference in performance between the attention-based and style-specific encoders in the sentiment-shift task may be explained by the fact that in the multi-encoder model for each input text, the encoder creates one static representation of input which is used only once to initialize the decoder. In the attention-based model, on the other hand, at each step of generation a latent representation of the input is created (which is aligned for that step of generation) and fed to the decoder. This reinforces the input content during generation and results in the presence of more input content in output (higher *CPP* scores).

In the domain of formality, on the other hand, we observe that the attention-based model, in spite of having high *SSP* score, does not converge to do TST effectively since it fails to meet the other requirements of the task such as preserving the content of inputs. The failure of the attention-based architecture to converge in the formality domain may be attributable to the smaller dataset in this domain as compared to the Yelp domain. We return to the challenge posed by small datasets for TST in chapter 7.

## 5.3 Sentiment versus content as compared with formality versus content

We conduct two experiments in this section to investigate the disentanglement of style and content and study how the relation between these two textual components vary across the domains of sentiment and formality while considering the baseline, multi-encoder and attention-based frameworks.

### 5.3.1 Probing the disentanglement of the style and content

Similar to the approach taken in section 4.3, we design a classification experiment to examine the latent representations of the input sequences created by the baseline, multi-encoder and attention-based models. To do so, we train a separate Feed-Forward networks with a single hidden layer and a sigmoid output layer as a probing classifier for each TST model and each style domain. These classifiers are trained to detect what the source style of a given input is.

Classifiers are trained using as input the latent vectors $\mathbf{z}$ and the input source style. $\mathbf{z}$ vectors fed to each classifier are created by the encoder of its corresponding TST model (figure 4.4). $\mathbf{z}$ vectors are considered as the last state of the encoder(s) for the baseline and multi-encoder systems. In the attention-based model a context vector is created for each generation step and we consider the latent representation of the input sequence as the average of these context vectors. To train these probes, within each domain of style, we merged the train and test sets and trained the probes using a cross validation technique (k-fold is set to 15).

The accuracy scores of the probing classifiers are reported in table 5.2 where higher scores indicate a higher presence of the source style in the latent space of its corresponding TST model. The high accuracy scores of the probes corresponding to the multi-encoder, 99.99%, and attention-based, 100%, TST architectures indicates that sentiment-content entanglement still holds while we modify latent space by employing different encoding strategies. Also, high accuracy scores of the probes corresponding to the multi-encoder, 99.6%, and attention-based, 100%, TST systems across the formality domain implies that that formality and content are also entangled elements. Finally, the average accuracy score in the sentiment domain is slightly higher (99.86%) than the average score reported for the formality domain (98.7%). This may indicate that sentiment is more entangled with content as compared with formality.

| Datasets | **Metrics** | **Baseline** | **Multi-encoder** | **Attention-based** | **Average** |
|---|---|---|---|---|---|
| ***Yelp-large*** | **Accuracy** | 99.58% | 99.99% | 100% | 99.86% |
| ***GYAFC-$v_2$*** | **Accuracy** | 96.5% | 99.6% | 100% | 98.7% |

Table 5.2 The accuracy of the probing classifiers (Accuracy) corresponding to each TST model across sentiment (*Yelp-large*) and formality domains (*GYAFC-$v_2$*).

To confirm that these scores are statistically different, we considered the accuracy scores of TST models across each domain of style and computed the confidence intervals of the scores. The results show that the scores of the three probes trained on *GYAFC-$v_2$* dataset do not have overlap with the confidence level of 0.67. This confidence level drops to 0.65 for the classifiers trained using the *Yelp-large* data.

### 5.3.2 The effect of stripping out style from the latent space of a TST system

The goal of this experiment is to examine the effects of removing the source stylistic features from the latent representation of a style-transfer framework on how well it preserves the content of the input sequences.

To do so, we consider a variational extension of the baseline model where the latent variable $z_s$ is sampled from the posterior distribution $\mathcal{N}(\mu_x, \sigma_x)$ for each input text $x_s$ (figure 5.3). The latent vector $z_s$ is constrained to draw a smooth distribution by measuring its KL-divergence with respect to a prior distribution $\mathcal{N}(0, I)$. We use an extra KL-divergence loss in addition to the task-specific loss, and optimize a linear interpolation of these two as the reconstruction loss of the generator blocks.

$$\mathscr{L}_{rec} = \quad -\log \Pr_E(\widetilde{\mathbf{x}}^{(s)}|\mathbf{x}^{(\mathbf{s})}) + \mathbf{D_{KL}}(\Pr_{\mathbf{E}}(\mathbf{z}|\mathbf{x},\mathbf{s})||\Pr(\mathbf{z})) \tag{5.6}$$

Here, we hypothesize that forcing the z vectors for the differently-styled sequences within one style domain to resemble a prior distribution will result in losing the source stylistic

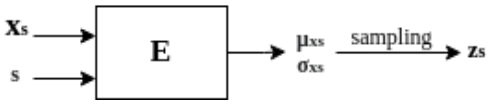


Figure 5.3 Variational extension of the baseline model where for each input $x_s$ latent vector $z_s$ is sampled from the posterior distribution $\mathcal{N}(\mu_x, \sigma_x)$.

features. Comparing the results of probing classifiers corresponding to the baseline model and its variational extension (table 5.3) confirms this hypothesis[4]. The results show that in both sentiment and formality domains, the probes of the variational models have lower accuracy in detecting the source style compared to the corresponding probes of the baseline models.

Considering the GloVe-based *CPP* scores and probing accuracy values corresponding to the baseline model and its variational extension (table 5.3), for each 1% drop of the accuracy of the probing classifiers, there is a *CPP* drop of 0.000259 in the formality domain as opposed to a *CPP* drop of 0.000454 in the sentiment domain. The higher drop of *CPP* in the sentiment as compared with formality domain suggests more entanglement between style and content in the domain of sentiment.

| Datasets | *Yelp-large* | | *GYAFC-$v_2$* | |
|---|---|---|---|---|
| Models | **Baseline** | **Variational** | **Baseline** | **Variational** |
| **Probing accuracy** | 99.58% | 44.46% | 96.5% | 51.41% |
| **CPP: GloVe-based** | 0.9239 | 0.8989 | 0.9239 | 0.8971 |

Table 5.3 Comparing the GloVe-based *CPP* and probing accuracy of the baseline model and its variational extension.

[4]To show that variational models are properly trained, we report *SSP* and fluency of the models. *SSP* across sentiment and formality domains are 96.98% and 71.01%. Fluency across sentiment and formality domains are 20.74 and 18.36.

**Further analysis** We compute the Pearson Correlation Coefficients (*PCC*) between the GloVe-based *CPP* scores of the baseline, multi-encoder and attention-based systems and the accuracy of the probing classification task using the corresponding system $\mathbf{z}$ representations (table 5.4) to further investigate the style-content entanglement across these two style domains. As the results indicate the *PCC* scores corresponding to the sentiment and formality domains are 0.699 and $-0.519$. These *PCC* scores compute the relation between the ability of a model to preserve the content of the inputs and how much source style is encoded in the latent vectors corresponding to their inputs, i.e. higher *PCC* scores show a stronger relation between the *CPP* ability of a system and its encoding of the style of the input texts. Therefore, the findings of this experiment imply more entanglement between style and content in the sentiment domain as compared to the formality domain.

To further investigate the validity of this observation, we examine whether using another *CPP* metric leads to similar *PCC* relations across formality and sentiment domains. To do so, we use the SBERT-based *CPP* metric (introduced in section 3.3.1.3) and recalculate the *PCC* scores considering probing classification scores and SBERT-based *CPP* scores corresponding to the baseline, multi-encoder and attention-based models (table 5.4). This leads to *PCC* scores of 0.701 and $-0.5536$ across the sentiment and formality domains, respectively. These *PCC* scores are aligned with the *PCC* scores computed using GloVe-based *CPP* metric which further confirm our proposition that sentiment and content are more entangled as compared with formality and content.

## 5.4 Discussion

In chapter 4, we observed that sentiment and content are overlapping components of the text. The main focus of the current chapter is to further study the latent space of the adversarial encoder-decoder RNN-based TST models to explore whether the observation from chapter 4 can be extended to other architectures and style domains. To do this, we conducted a

| Datasets | Metrics | Baseline | Multi-encoder | Attention-based | *PCC* |
|---|---|---|---|---|---|
| ***Yelp-****large* (sentiment) | **Accuracy** | 99.58% | 99.99% | 100% | - |
| | ***S-BERT*** | 0.3195 | 0.3613 | 0.4933 | 0.701 |
| | ***GloVe*** | 0.9239 | 0.9311 | 0.9542 | 0.699 |
| ***GYAFC-****$v_2$* (formality) | **Accuracy** | 96.5% | 99.6% | 100% | - |
| | ***S-BERT*** | 0.2408 | 0.2459 | 0.1733 | -0.536 |
| | ***GloVe*** | 0.9088 | 0.911 | 0.8869 | -0.519 |

Table 5.4 Pearson correlation coefficients (*PCC*) scores between the accuracy scores of the probing classifiers corresponding to the TST models, and both their GloVe-based *CPP* scores and SBERT-based *CPP* scores across the two domains of formality and sentiment.

classification experiment to probe the latent space of the TST models. The current chapter also examines the consistency of the concept of style considering the sentiment and formality domains. To do so, we designed some experiments to explore the sentiment-content versus formality-content levels of entanglement. Throughout these experiments, we used three models which have similar architecture but different encoding techniques which can affect the latent space. Modifying the latent space enables us to better explore the latent space. The main findings of this chapter are as follows.

- The entanglement of sentiment and content still holds while we modify the latent space by employing different encoding strategies. This is inferred from the high accuracy of the probes in sentiment domain (row 1 of table 5.2).

- The observation of style-content entanglement can be extended to other domains of style since the high accuracy scores of the probing classifiers indicated that formality and content are also overlapping elements (row 2 of table 5.2).

- The concept of style is not consistent across the sentiment and formality domains, i.e. sentiment and content are more entangled as compared with formality and content. The following observations support this finding. Firstly, the average accuracy score of the sentiment probes is 99.86% which is slightly higher than 98.7%, the average accuracy score of the formality probes (table 5.2). Also, the results of table 5.3 indicate

that variational frameworks encode less source style in their latent space compared to baseline models across the both style domains. Taking these accuracy scores and *CPP* scores of the models together (table 5.3) indicates that for each 1% drop of the accuracy of the classifiers, there is more drop of *CPP* in the sentiment domain as compared with the formality domain which implies more entanglement between sentiment and content. Furthermore, the *PCC* scores between the GloVe-based *CPP* scores of the baseline, multi-encoder and attention-based systems and their accuracy of the probing classification task (table 5.4) indicate a higher level of entanglement between the sentiment and content (0.699) as compared to formality and content ($-0.519$).

- We explored the TST task across sentiment and formality domains by studying the performance of the baseline model, multi-encoder and attention-based systems. The results (table 5.1) shows a trade-off between the evaluation aspects of the TST task, an inverse relation between *CPP* scores and both *SSP* and fluency values of the models[5]. This trade-off which is inline with the trade-off which was earlier observed in section 4.5 further implies the necessity of employing comprehensive evaluation methodologies while dealing with the TST problem.

The findings of this chapter which are listed here address the research questions (section 1.2) of this manuscript as follows. Findings 1 and 2 address research question 1. Findings 3 and 4 address the research questions 2 and 5.

## 5.5 Conclusion

We observed in chapter 4 that sentiment and content cannot be totally separated. The results of the current experiments not only indicate that sentiment and content are overlapping elements, but they showed that this observation also holds across other architectures as well

[5]The results of the attention-based in the formality domain are excluded while doing the analysis in this section, since this model has not converged to do the task effectively.

as the style domain of formality. Moreover, the results revealed that sentiment and content are more entangled as compared to formality and content. These findings imply that the concept of style can differ from one domain to another at least in terms of how it relates to content.

Observing that a single concept of style cannot be generalisable across the domains, firstly, highlights the question of what are the variations across the style domains and how these variations are encoded. Secondly, it raises the question of whether style-specific characteristics should be taken into consideration when designing a TST experiment, i.e. for instance if some architectures work better in one style domain due to the characteristics of that domain. In the next chapter, we design a series of experiments to investigate these questions.

| **From negative to positive** |
|---|
| 1. so nasty . |
| 2. so nice . |
| 3. so delicious . |
| 4. so delicious . |
| |
| 1. my goodness it was so gross . |
| 2. my delicious it was delicious . |
| 3. my goodness it was so delicious . |
| 4. my goodness it was amazing . |
| |
| 1. fries are n't worth coming back . |
| 2. fries are worth coming back ! |
| 3. fries are n't worth coming back . |
| 4. wings are coming back . |
| |
| 1. the cake portion was extremely light and a bit dry . |
| 2. the light and very light with a bit old . |
| 3. the filet are extremely dry and a bit dry . |
| 4. the tuna was a bit dry and satisfying . |
| **From positive to negative** |
| 1. i highly recommend this place ! |
| 2. i would not recommend this place ! |
| 3. i would not recommend this place . |
| 4. i do not recommend this place ! |
| |
| 1. they had a salmon dinner on special and it was outstanding ! |
| 2. they had a few minutes with dinner dinner and it was horrible ! |
| 3. they had a dinner dinner on special and it was outstanding ! |
| 4. they had a reservation dinner and it was out on it was cold ! |
| |
| 1. my appetizer was also very good and unique . |
| 2. my appetizer was also very good and tasted weird . |
| 3. my appetizer was also very good and unique . |
| 4. my boyfriend was n't and had a very dry . |
| |
| 1. the food is fresh and the environment is good . |
| 2. the food is fresh and the sushi is very poor . |
| 3. the food is fresh and the environment is good . |
| 4. the food is fresh and the sandwich was too salty . |

Table 5.5 Style-shifted outputs of the TST models using *Yelp-large*. Sequences are ranked as: 1: Input sequence, 2: Multi-encoder model, and 3: Attention-based model 4: Base model.

| **From informal to formal** |
|---|
| 1. be cool and ask her on a date ! |
| 2. be yourself , you get her own ! |
| 3. go to go for a own ! |
| 4. and just get out with a mind ! |
| |
| 1. ask him to go see a doc . |
| 2. ask him to get a lot . |
| 3. you go to go out . |
| 4. ask him to go to a doctor . |
| |
| 1. the show got a little freaky . |
| 2. the movie has a lot . |
| 3. the movie is a great . |
| 4. the movie has a little friend . |
| |
| 1. do you mean seem more sexy ? |
| 2. do you you have sex ? |
| 3. do you have anything ? |
| 4. do you mean more attractive ? |
| **From formal to informal** |
| 1. what should i do to be successful with her ? |
| 2. what can i want to be with her ? |
| 3. why does n't get to get with you ? |
| 4. what i do n't want to be with him ? |
| |
| 1. i am assuming that you are a republican . |
| 2. i am sure you are a cheater . |
| 3. i think it is not a problem . |
| 4. i am sure that he is a jerk |
| |
| 1. you need to do this a few times to get comfortable . |
| 2. you need to do a lot of them to be married . |
| 3. you should do n't get to get a chance . |
| 4. you do a lot to go to be more time . |
| |
| 1. make it an honest compliment about her . |
| 2. try it for her for her . |
| 3. it depends on you . |
| 4. get a <unk> and tell him this . |

Table 5.6 Style-shifted outputs of the TST models using *GYAFC-$v_2$*. Sequences are ranked as: 1: Input sequence, 2: Multi-encoder model, and 3: Attention-based model 4: Base model.

# Chapter 6

# Style: locally or globally encoded?

The findings of the previous experiments (chapters 4 and 5) indicated that the concept of style is not consistent across the sentiment and formality domains. In this chapter, we design a series of experiments to examine how these variations are encoded. Firstly, we look into an adversarial transformer-based encoder-decoder TST system and explore the input latent representations created by different layers of this architecture considering the sentiment and formality domains. Our motivation for employing transformer-based TST systems as opposed to single-layered RNNs is that different layers of transformers learn to encode different textual information according to the previous research (Nedumpozhimana and Kelleher, 2021; Nedumpozhimana et al., 2022). It is interesting to examine how transformer layers encode stylistic features across different style domains while dealing with the TST problem. Furthermore, we design an experiment to compare the effect of modifying the weight of the reconstruction loss of the transformer-based TST model on shifting the sentiment versus shifting the formality. The idea being that how these models perform considering specifically the style-shift dimension can further inform the variations between the style domains. We take a further step towards studying the variations between different style domains through a unigram analysis experiment where in addition to formality and sentiment, this experiment also uses the simplicity style domain.

Finally, we investigate whether variations across the style domains should be considered while framing the TST task. We specifically look into how style characteristics can inform the strategy that we adopt to compute the *CPP* of a TST framework.

Throughout the experiments of this chapter, we use the *Yelp-small* corpus (described in section 3.1.1), *GYAFC-$v_2$* corpus (described in section 3.1.2) and *Newsla-$v_2$* corpus (described in section 3.1.3) to study the style domains of sentiment, formality and simplicity. To evaluate the performance of the transformer-based TST frameworks, we use the comprehensive evaluation methodology already applied in the previous chapters to examine the RNN-based TST systems.

## 6.1 The proposed transformer-based TST model

Due to the success in many NLP tasks in recent years, including machine translation (Vaswani et al., 2017), and language modelling (Dai et al., 2019c), transformers (Vaswani et al., 2017) have attracted the attention of many NLP researchers. TST researchers have also started to develop transformer-based (T-based) TST models which have outperformed some state-of-the-art RNN-based models (table 6.1, see rows 1 and 2 versus row 3). In this chapter, we also employ T-based models and base our experiments on this architecture.

We begin by introducing our proposed T-based model[1] which is similar to the T-based baseline model. The T-based baseline model (introduced in section 3.2.3) is based on the model proposed by Dai et al. (2019b). We will refer to our proposed T-based model as the "proposed T-base model" and to the model introduced by Dai et al. (2019b) as the "T-based baseline model". The proposed T-based model includes an encoder-decoder network as the Generator block ***Gen*** as well as a separate classifier as a Discriminator block ***Disc*** (similar to the T-based baseline model). However, it applies an adaptation to the training regime of the

[1]The code and data are available at https://github.com/somayeJ/Transformer-based-style-transfer.git

**Algorithm 3: Adversarial training of our T-based model**

**Input:** ***Gen***$(\theta_E, \theta_D)$, ***Disc*** $(\theta_{disc})$ and any two corpora $X_{s_1}$ and $X_{s_2}$ which have the same content distribution but different styles $(s_1 \neq s_2)$.

**1.** Sampling two mini-batches with the size $k$ from the sets $X_{s_1}$ and $X_{s_2}$ (setting $k = 1$ for the sake of simplicity).
**2.** Processing the two mini-batches in parallel, i.e. for each of the sequences $x_{s_1} \in X_{s_1}$, and $x_{s_2} \in X_{s_2}$, ***Gen*** generates a reconstructed and a style-shifted sequence for each input:

- For $x_1$: $\widetilde{\mathbf{x}}_{s_1}^{(rec)} = G(x_{s_1}, s_1)$
  $\widetilde{\mathbf{x}}_{s_1}^{(trf)} = G(x_{s_1}, s_2)$
- For $x_2$: $\widetilde{\mathbf{x}}_{s_2}^{(rec)} = G(x_{s_2}, s_2)$
  $\widetilde{\mathbf{x}}_{s_2}^{(trf)} = G(x_{s_2}, s_1)$

**3.** Computing the self-reconstruction loss using the equation 3.8.
**4.** Computing the discriminator loss using the equation 3.7.
**5.** Computing the cycle-reconstruction loss using the equation 3.9 and adversarial loss using the equation 3.11.
**6.** Training ***Disc*** by doing step 4 and performing gradient decent to update $\theta_{Disc}$ using gold, reconstructed and style-shifted sequences.
**7.** Checking the condition $\mathscr{L}_{Disc} < 1.2$ (a pre-set threshold set to 1.2 following Shen et al. (2017)).

**-** If $\mathscr{L}_{Disc} < 1.2$: Doing steps 3 and 5 , reconstructed and style-shifted sequences. Performing gradient decent to update $\theta_{Gen}$.

**-** Otherwise: Doing only step 3 and performing the backpropagation and updating $\theta_{Gen}$.

**8.** For all the epochs (20 here), repeating steps 1, 2 , 6 and 7 for all batches and selecting the model with lowest total loss (equation 3.12) as the best model after each evaluation step, i.e. after each epoch.

T-based baseline model which is explained in the *Algorithm 3: Adversarial training of the proposed T-based model*.

The training regime used in the T-based baseline model (Dai et al., 2019b) which is described in details in section 3.2.3 differs from the training regime of the proposed T-based model in a number of ways:

- T-based baseline model has a pre-training step to only train ***Gen*** for a number of batches[2] using just the self-reconstruction loss. However, the training of the proposed

[2]The number of batches used in this pre-training is a hyperparameter.

T-based model does not involve any pre-training steps, i.e. the whole model is directly trained from scratch.

- T-based baseline model starts the training by alternating between training ***Gen*** and ***Disc*** where, for a number of batches, ***Gen*** and then, for a number of batches ***Disc*** is trained[3]. However, in the proposed T-based model ***Gen*** and ***Disc*** are trained together in parallel. The adversarial loss is not used in this training regime until ***Disc*** has sufficient quality that the adversarial loss it returns is informative.

- In each step of training of the T-based baseline model, two processes of parameter updates are involved. First, backpropagation of self-reconstruction loss and updating the weights accordingly. Then, backpropagation of the summation of the adversarial and cycle loss, followed by weight updates. In the proposed T-based model, on the other hand, the backpropagation and the weight updates of the parameters are performed once using the summation of adversarial loss, and the two reconstruction losses.

**Experimental setup of the proposed T-based TST model** Each stack of ***E*** and ***D*** of our T-based model has 4 attention heads. The size of token embeddings, positional embeddings, style vectors and the hidden size of the model are 256. The max-length for generated outputs is 15 for Yelp and 24 GYAFC and the learning-rate is 0.0001 for both *Gen* and *Disc*. The evaluation-step is 1 epoch, the optimizer is ADAM, the batch-size is 64 and the dropout-rate is 0[4].

**The performance of the proposed T-based model** Table 6.1 reports the results of the proposed T-based model, the baseline T-based model and the RNN-based baseline model. The results show that the proposed T-based model outperforms our state-of-the-art RNN-

[3]The number of batches used in training ***Gen*** and ***Disc*** is another hyperparameter.

[4]Hyperparameters of the proposed T-based model are mostly adapted from (Dai et al., 2019b) (http:/github.com/fastnlp/style-transformer) and are explained in in table 3.9.

| **Dataset** | ***Yelp-small* (*sentiment*)** | | | ***GYAFC-$v_2$* (*formality*)** | | |
|---|---|---|---|---|---|---|
| **Model/ Evaluation metrcis** | **CPP** | **PPL** | **SSP** | **CPP** | **PPL** | **SSP** |
| T-based baseline model* | 0.9717 | 106.07 | 78.50% | 0.9516 | 289.20 | 28.99% |
| Proposed T-based model | **0.9718** | 126.12 | **83.00 %** | **0.9741** | 141.44 | 47.19% |
| RNN-based baseline model | 0.9261 | **37.98** | 81.8% | 0.9088 | 26.81 | **65.11%** |

* The code released by Dai et al. (2019b) did not specify a training stopping criterion and so to reproduce these results we specified our own stop-criterion and model selection strategy (see algorithm 2 in section 3.2.3.3).

Table 6.1 Higher *CPP* and *SSP* show better performance, but lower values of *PPL* reflect better fluency. $\alpha$ and $\beta$ of the reconstruction loss of the both T-based models are set to 0.25 & 0.5 (equation 3.10).

based baseline model (proposed by Shen et al. (2017)) especially in terms of *CPP* (see rows 2 and 3). Also, comparing rows 1 and 2 of table 6.1 shows that the proposed T-based model slightly improves the performance of T-based baseline model, a slight improvement in CPP and SSP on Yelp and a larger improvement in CPP for GYAFC with a drop in SSP. However, these results are recorded from single runs of the model and so we do not claim a statistical difference here.

More importantly, however, applying the adapted training regime reduces the computational cost, i.e. to reach these results, the training time needed for the the proposed T-based model is lower than the training time needed for the baseline model. To be more precise, it took around 36 hours to train the proposed T-based model as compared to 75 hours training time applying the training regime from Dai et al. (2019b) while using the same hardware (single Quadro RTX 8000s GPU) and corpus (Yelp-*small*)[5].

## 6.2 Reconstruction loss versus adversarial loss

In the next experiment, we investigate the variations between sentiment and formality domains by modifying the weight of reconstruction loss while keeping the weight of the adversarial loss constant. The total loss of the proposed T-based model is the weighted

[5]The number of trainable parameters of the T-based model is 19859513.

| | Datasets | | *Yelp-small* | | *GYAFC-$v_2$* | |
|---|---|---|---|---|---|---|
| | Models | | $T_{rec}$ | $T_{cyc}$ | $T_{rec}$ | $T_{cyc}$ |
| Evaluation Metrics | **SSP** | | **83.8%** | 70.9% | 32.71% | **41.52%** |
| | **PPL** | | 107.07 | **99.88** | **101.57** | 154.35 |
| | **CPP** | GloVe | 0.9732 | **0.9767** | **0.9743** | 0.9714 |
| | | GloVe-LB | 0.86 | | 0.87 | |
| | | SBERT | 0.5869 | **0.6177** | **0.8595** | 0.8108 |
| | | SBERT-LB | 0.0939 | | 0.0672 | |
| | | WO | 0.5728 | **0.6305** | **0.8154** | 0.7357 |
| | | WO-LB | 0.0101 | | 0.0045 | |

Table 6.2 $\alpha$ and $\beta$ of the proposed T-based models $T_i$ are: ($T_{rec}$; $\alpha$=1, $\beta$= 0.5), ($T_{cyc}$; $\alpha$=0.5, $\beta$=1), *LB* indicate the Lower Bound score. ($\alpha$ and $\beta$ shown in the equation3.10)

summation of the self-reconstruction, cycle-loss and adversarial loss (equation 3.12) and the contribution of these three losses are not normalized. Therefore, increasing the weights of the self-reconstruction ($\alpha$ in 3.10) and cycle-reconstruction ($\beta$ in 3.10) can push the TST model to behave more as an auto-encoder. Re-weighting of the reconstruction losses of the TST model is an interesting experiment to reveal the characteristics of different style domains. The idea here is that the more a TST model is weighted towards acting as an auto-encoder, the less it is able to shift the style. This can be specifically reflected in the case of dealing with styles that are pervasive across a text and which the rewriting of a text into requires global modifications to the text.

In the previous experiments (results of table 6.1) models were trained with a relatively large emphasis on the adversarial loss during training, since the summation of the weights of the self-reconstruction $\alpha = 0.25$, and cycle-reconstruction $\beta = 0.5$ equal to 0.75 which is less than 1 which is the weight of adversarial loss. In this experiment, we train the proposed T-based model with a greater emphasis on the reconstruction loss as opposed to adversarial loss by doubling the summation of $\alpha$ and $\beta$ and train two models by having the weight of reconstruction loss equal to 1.5 as opposed to adversarial weight which is kept constant as 1. We train two new models: $T_{rec}$; $\alpha$=1, $\beta$= 0.5, and $T_{cyc}$; $\alpha$=0.5, $\beta$=1. Comparing the results of $T_{rec}$ and $T_{cyc}$ in table 6.2 with the scores of the proposed T-based

model (row 2 of table 6.1) indicates that, in Yelp, $T_{rec}$ performs better than the proposed T-based model in every evaluation aspect and $T_{cyc}$ also has a better *CPP* and fluency. In GYAFC-$v_2$, however, increasing the weighting towards reconstruction loss does not appear to be as beneficial overall. The performance of $T_{cyc}$ drops in every aspect of evaluation and although an improvement is observed in *CPP* and fluency for $T_{rec}$, the *SSP* for $T_{rec}$ drops by a large amount.

Increasing the weight of the reconstruction loss relative to the adversarial loss is beneficial for both *CPP* and *SSP* in the sentiment domain but results in much lower *SSP* in the formality domain. This suggests that shifting style in the sentiment domain requires fewer text changes compared to the formality, i.e. sentiment is more locally encoded compared to formality.

During training, the re-weighted TST models tended to act more similarly to an auto-encoder while using the GYAFC-$v_2$ corpus compared to when Yelp-*small* was used. For instance, we trained $T_{rec}$ a number of times from scratch so that it converged as a TST model and reached the reported performance in table 6.2. When it failed to converge, this model kept reaching very low *SSP* scores, an average of 15%[6] and very high *CPP* scores, an average GloVe-based *CPP* score of 0.991, almost the same as GloVe-based *CPP* scores of the reconstructed files. These results which are very similar to when the model is only trained to reconstruct inputs (auto-encoder) suggest that decreasing the relative emphasis on adversarial loss as compared to the reconstruction loss results in the models finding it more difficult to learn how to shift formality. Tables 6.8 and 6.9 list some samples of style-shifted outputs created by the TST models. These samples also show how T-based models act more like auto-encoders in formality domain as compared with the sentiment domain.

[6]As table 3.10 shows, the lower bound score of *SSP* in GYAFC-$v_2$ dataset is 13.89%.

## 6.3 Encoding variations across sentiment and formality domains

We design some experiments here to investigate the variations of style domains by firstly looking into how different layers of the proposed T-based encoder of the proposed TST model encode different styles (section 6.3.1). Then, we compare the observations from this experiment with the variations that can be detected in human-generated data across the style domains (section 6.3.2).

### 6.3.1 Probing the layers of encoders of T-based models

An interesting aspect of transformer models is that they include multiple self-attention layers. Indeed, researchers interested in understanding how transformers encode linguistic information have probed how the encoding of this information varies across the layers of transformers trained for different NLP problems (Nedumpozhimana and Kelleher, 2021; Nedumpozhimana et al., 2022). However, to the best of our knowledge, the encoding of style across the layers of a TST transformer has not yet been examined. This experiment focuses on examining different layers of the encoder of the proposed T-based TST model and comparing how these layers encode formality and sentiment. Specifically, this experiment investigates the presence of source style in the input latent representations created by each layer of the encoder of the proposed T-based TST system.

To do so, inspired by some previous work (Conneau et al., 2018; Jafaritazehjani et al., 2020, 2021), we design a probing classification experiment and train 6 probes (classifiers), i.e. a probe for embedding layer, 4 separate probes for the outputs of each layer of the encoder subnetwork, and a probe for the final output of the encoder (figure 6.1). Each probe is a Feed-Forward network with a single hidden layer and a sigmoid output layer and is trained to detect the source style of the embedding representation of the input text which is created

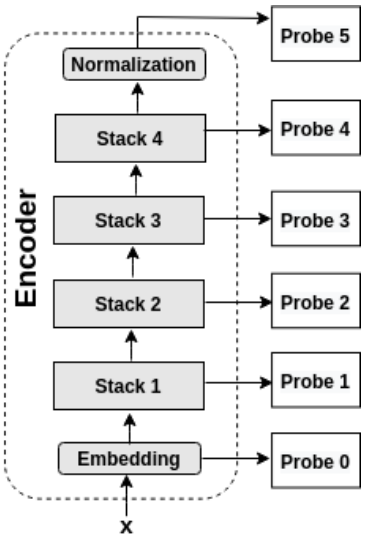


Figure 6.1 Probing classification experiment on different layers of transformer encoder

by the layer corresponding to that probe. This means that the higher the accuracy score of a probe, the more source style is encoded in that layer. Given an input sequence, the embedding vector created by a layer is computed as the average of the token embeddings of this sequence generated by that layer.

Moreover, we train two baseline probes, one for each style domain. To do so, we create embedding vectors of the data using a pre-trained GloVe model, i.e. given a sequence, first, its tokens are mapped to their 100-dimensional GloVe embeddings. The average of these embeddings is then computed as the sequence embedding. The binary baseline probes for the sentiment and formality domains are trained using GloVe-based representation of the Yelp and GYAFC data, respectively.

The accuracy of the baseline probes across sentiment and formality domains are 0.85% and 0.71% respectively. This indicates that sentiment information are better encoded using a bag-of-words based pre-trained embedding model compared to formality. GloVe embedding are trained considering word-word co-occurrence and disregarding word positions and their order in a sequence (Pennington et al., 2014). This makes GloVe-based embedding vectors created in this experiment for each sequence a bag-of-words representation focusing more

| | Datasets | *Yelp-small* | | *GYAFC-$v_2$* | |
|---|---|---|---|---|---|
| | Models | $T_{rec}$ | $T_{cyc}$ | $T_{rec}$ | $T_{cyc}$ |
| | **GloVe-baseline** | 85.80% | | 71.01% | |
| Layer-wise probing of encoder of the proposed T-based model | **Embedding layer** | 89.9% | 87.4% | 75.74% | 78.69% |
| | **Stack1** | 100% | 90.5% | 74.18% | 80.48% |
| | **Stack2** | 100% | 100% | 99.63% | 88.2% |
| | **Stack3** | 100% | 100% | 100% | 100% |
| | **Stack4** | 100% | 100% | 100% | 100% |
| | **Final output** | 100% | 100% | 100% | 100% |

Table 6.3 $\alpha$ and $\beta$ (equation3.10) of the proposed T-based models $T_i$ are: ($T_{rec}$; $\alpha$=1, $\beta$= 0.5), ($T_{cyc}$; $\alpha$=0.5, $\beta$=1), *LB* indicate the Lower Bound score.

on the local information of the tokens. On the contrary, it can be said that sequence vectors created using the outputs of different layers of the transformer encoder consider the contextual and positional information of the tokens . This is due to the attention-based architecture of this model which allows the token embeddings to be fine tuned with regards to its surrounding tokens which leads to encoding more contextual-based information. Adding more stacks to the encoder may result in better encoding of the information of the context of use of the tokens. Therefore, the higher the stack of the encoder is the more global information it encodes.

The probing experiment in this section studies how style-related information is encoded in different stacks of the encoder across the domains of sentiment and formality. Table 6.3 show the results of layer-wise probing of the encoder of T-based model. These results illustrate that the probes of sentiment domain reach 100% accuracy in lower layers compared to the probes of formality domain. This indicates that formality is more sensitive towards global structure of the sentence, i.e. there is a need to encode more information from across the sequence to create the token embedding representations in formality as compared with sentiment.

The results of this probing experiment suggest that sentiment is more locally encoded compared formality. This means that sentiment is more reliant on local information, for

instance tokens can be considered as style bearing segments of sequences. However, in the domain of formality, style is more globally encoded which makes it more reliant of structural information of sequences. In the next section, we examine whether this hypothesis holds by doing a unigram analysis across these two domains.

### 6.3.2 Unigram analysis

Results of the layer-wise probing of the transformer encoder (section 6.3.1) indicate variations in how sentiment and formality styles are encoded. These results suggest that sentiment is a more local entity of the text as opposed to formality which is more global. To further investigate this, we designed a unigram-based experiment. The idea being that it can enable us to compare how many word swaps must be done in a style domain so that style-shift to the extent of *SSP* of the style-shifted text happens considering the domains of sentiment, and formality. We then extended this experiment to the style domain of simplicity too.

This experiment focuses on comparing the token overlaps of inputs and style-shifted gold outputs created by human across different style domains[7]. Word overlap between two texts is calculated as the unigram overlap rate between them[8]. In this experiment, for the dataset Yelp-*small*, we compute the word overlap between the sequences of the test set and their corresponding human-generated (gold) style-shifted texts and report the average score of these pairs as the word overlap score of these two sets. For the dataset GYAFC-$v_2$, for a given test file of style 1 (either formal or informal) of each of the domains of Entertainment & Music (E&M) or Family & Relationships (F&R), there are 4 gold style-shifted files. The word overlap for each domain is calculated as $\frac{(\sum_{s=1}^{2}\sum_{i=1}^{4} WO_{si})}{8}$ where $WO_{si}$ is the word overlap between the test file of style $s$ and gold style-shifted file $i$. The average of the word overlap of the two domains is reported as the word overlap score of the dataset GYAFC-$v_2$ (table 6.4).

[7]We took a further step and did this experiment while comparing test data with the style-shifted outputs of the of the RNN-based TST models of baseline, attention-based and multi-encoder instead of gold human-generated style-shifted sequences. The two experiments led to similar results and conclusions.

[8]This metric is explained in section 3.3.1.

| | Metrics | Average scores of metrics | |
|---|---|---|---|
| | | Word overlap | Accuracy |
| Datasets | Yelp-*small* (sentiment) | 0.4253 | 77.20% |
| | GYAFC-$v_2$ (formality) | 0.4057 | 70.45% |
| | Newsla-$v_1$ (simplicity) | 0.3615 | 79.2% |

Table 6.4 The results of word overlap between sequences of test set and their gold style-shifted text and accuracy of detecting desired style of gold style-shifted text.

We also compute the percentage of the presence of the desired style in the gold style-shifted sets corresponding to the test sets of the datasets Yelp-*small* and GYAFC-$v_2$ as an indicator of how well the annotators have shifted the styles of the test sets. To do so, for each dataset, Yelp-*small* and GYAFC-$v_2$, the accuracy of its corresponding pre-trained *SSP* classifier[9] in detecting the desired style of the gold files is measured (column *Accuracy* of table 6.4). For the domain of formality where several gold style-shifted sets are available, for each domain, the reported score is the average of accuracy values that the pre-trained classifier assigns to each domain which is computed as $\frac{(\sum_{s=1}^{2}\sum_{i=1}^{4} ACC_{si})}{8}$ where $ACC_{si}$ is the accuracy of the classifier in detecting labels of the gold text in file $i$ with the source style $s$.

The results of rows 1 and 2 of table 6.4 shows that, in the domain of sentiment, there is more word overlap between the test files and their corresponding gold style-shifted text 0.4253 as compared with the formality domain 0.4057. The difference between the word overlap values 0.02 is meaningful since the lower bound of word overlap is 0.0045[10]. This taken together with the higher accuracy of the pre-trained *SSP* classifiers in detecting the desired labels of the gold files of the sentiment domain (77.20% as opposed to 70.45% in formality domain) further validates our hypothesis of sentiment-shift being more reliant on local changes and a relatively small number of word swaps as compared with formality which seems to be a more global entity. Observing a higher word overlap as well as a higher *SSP* in sentiment domain versus the formality domain suggests that, to shift formality effectively,

[9]Pre-trained *SSP* classifiers are introduced in section 3.3.1.

[10]How lower bounds are computed is explained in section 3.3.1.

there is a need to do more than word swaps. This can further indicate that formality-shift is a more complex task as opposed to sentiment-shift.

**Investigating the *simplicity* style domain** To better contextualise these results, we extend our analysis to the style domain of *simplicity* using dataset Newsela-$v_1$ (described in section 3.1.3) which contains sequences in styles *complex* and *simple*. The results of row 3 of table 6.4 show that the *simplicity* domain exhibits the lowest word overlap of the three domains 0.3615. Interestingly, although the simplicity domains also has the highest style-shift accuracy score (SSP of 79.2%) across the three domains, this accuracy score is quite close to that accuracy score of the sentiment domain (SSP of 77.2%). This is somewhat surprising as one might expect that given the much lower word overlap in simplicity versus sentiment (0.3615 versus 0.4253) that this would result in a much larger difference in accuracy across the domains. One way to interpret this is that the concept of style in the simplicity domain relies on more than differences in words but also on structural properties of the text (i.e., simplicity-transfer involves both token-based swaps and structure-based changes) and that modellings these structure based changes is challenging for the style classifiers, hence the lower than expected accuracy of these classifiers in the simplicity domain. Consequently, this may suggest a stronger style encoding similarity exists between *simplicity* and *formality* than *simplicity* and *sentiment*.

## 6.4 The intersection between style characteristics and TST task

Overall the results of the experiments of the current chapter indicate that sentiment is a more local phenomenon within a text as compared with formality. This section investigates whether these style domain specific characteristics should be considered while framing the

TST task. It first examines the evaluation metric of *CPP* by looking into the performance of GloVe-based versus SBERT-based *CPP* metrics across the sentiment and formality domains (section 6.4.1). Then, it discusses the performance of various architectures while dealing with sentiment-shift and formality-shift (section 6.4.2).

### 6.4.1 SBERT-based versus GloVe-based *CPP* metric

One of the common approaches to compute the *CPP* of TST models is to rely on cosine similarity technique (section 3.3.1) where different pre-trained embedding models can be applied to create token embeddings. However, many previous work such as (Fu et al., 2018a; Shen et al., 2017; John et al., 2019; Jafaritazehjani et al., 2020, 2021, 2022) used a GloVe pre-trained embedding model to map tokens to their pre-trained embedding vectors.

We observed in section 6.3 that encoding the information in some style domains such as formality is beyond the representational capacity of a bag-of-words and is more a global property of the text. This insight questions the suitability of applying pre-trained embedding models, such as GloVe, which focus more on local information in text to compute token embeddings. Here, we propose to compute *CPP* using contextual embeddings by applying the pre-trained model of SBERT[11] (Reimers and Gurevych, 2019). SBERT is a T-based architecture which, given a text, applies a layered attention mechanism to create embeddings of tokens which leads to capturing more of the structural and global properties of a text. Computing SBERT-based *CPP* scores enables us to compare the performance of pre-trained contextual embedding models versus pre-trained bag-or-words embedding models in capturing the information of the text across the sentiment and formality domains.

**SBERT-based versus GloVe-based *CPP* across different style domain using human-generated text** To examine the performance of the SBERT-based versus GloVe-based *CPP* metrics across sentiment and formality domains, we conduct an experiment as follows.

[11] https://huggingface.co/cross-encoder/stsb-TinyBERT-L-4

| | *CPP* **Metrics** | GloVe-based | SBERT-based |
|---|---|---|---|
| Datasets | Yelp-*small* (sentiment) | 84% | 75.5% |
| | GYAFC-$v_2$ (formality) | 71% | 95.5% |

Table 6.5 Comparing the performance of GloVe-based and SBERT-based *CPP* metrics in assigning the highest scores to the most similar pairs across the domains of sentiment and formality

For each style domain, we randomly select 200 samples from the test set of that domain as the source sequences, i.e. 100 samples from the style 1 and 100 samples from style 2. For instance, in the domain of formality, 100 formal samples and 100 informal sequences are randomly selected. For each of the selected source sequences, we then create a style-shifted set which contains its corresponding gold style-shifted sequence[12] as well as 499 other sequences which are randomly selected from another domain of style. This means that, if we are performing the experiment in the domain of sentiment, we select the random sequences from another domain such as formality. This is to reduce the chances of the random sequences resembling the output which may occur in the data with the same content distribution.

After composing the source set and style-shifted set, we compute the *CPP* scores between each given source sequence and the sequences in its corresponding style-shifted set using both SBERT-based and GloVe-based metrics. We expect that, given a source text, a good *CPP* metric assigns a higher value to the pair of (source text, its corresponding style-shifted text) rather than to the pairs of (source text, random text 1), ..., (source text, random text 499). The results of table 6.5 show that SBERT-based metric performs better in the formality domain by assigning the highest values to the pair of (source text, its corresponding style-shifted text) in 95.5% of the times as compared to the 75.5% of times in sentiment domain. This implies that contextual embeddings of SBERT models are more efficient to compute *CPP* scores in the formality domain which is inline with our observation suggesting that formality

[12]In the domain of formality, there are four gold sequences (four human annotators) available for each test sequence. The gold samples are randomly selected from these four sets.

is encoded as a global property of a text, i.e. it is beyond the representational capacity of a bag-of-words.

On the other hand, the scores of table 6.5 indicate that the GloVe-based *CPP* metric does a more precise job in the sentiment domain by assigning the highest values to the pair of (source text, its corresponding style-shifted text) in 84% of the times as compared to 71% of the times in the formality domain. As mentioned earlier, it seems like while computing the similarity of the sequences, GloVe-based metric assigns high scores to the pairs of (source text, its corresponding style-shifted text) which resemble in content regardless of their different sentiment (table 6.6 sentence pairs 1 and 2). However, SBERT-based *CPP* metric assigns low scores to these pairs which indicates that SBERT embedding model differentiates between tokens having different sentiment. This can shed more light on why GloVe-based *CPP* metric compute more precise scores in the sentiment domain compared to the SBERT-based metric[13].

**Score variations of SBERT-based versus GloVe-based** *CPP* Firstly, comparing the SBERT-based *CPP* scores of the table 6.2 and the GloVe-based scores of the tables 4.1, 5.1, 6.1, and 6.2 illustrates that SBERT-based values have a larger range compared to GloVe-based scores which are quite similar in values.

Given the upper bound 1 of a cosine similarity based metric, the range for a given metric and a given domain is computed as the difference between the upper and lower bound scores of the metric in that domain. The range for the GloVe-based scores in sentiment and formality domains are 0.14 and 0.13 respectively. However, for the SBERT-based values, these ranges changes to 0.9061 and 0.9328 for sentiment and formality domains (lower bounds of these metrics are reported in table 6.2). The larger variation range of the SBERT-based metric

[13]We extended the experiment by investigating how SBERT-based and GloVe-based metrics compute the *CPP* scores considering the style-shifted outputs of of the RNN-based TST models of baseline, variational, attention-based and multi-encoder instead of the gold style-shifted sequences. The results showed that the findings are valid while using automatically generated style-shifted text.

| | *Samples from sentiment domain* | *SBERT* | GloVe |
|---|---|---|---|
| 1 | s: there is definitely not enough room in that part of the venue .<br>t: there is definitely enough room in that part of the venue . | 0.26 | 0.999 |
| 2 | s: ever since he has changed hands it's just gotten worse and worse .<br>t: ever since he has changed hands it's always gotten cool and cool . | 0.41 | 0.988 |
| | *Samples from formality domain* | *SBERT* | GloVe |
| 3 | s: u mean the question to *_num_* question right ?<br>t: it is the question to *_num_* . | 0.56 | 0.894 |
| 4 | s: thats weird but maybe idk ill talk to my friend by his locker .<br>t: it is but maybe two . | 0.17 | 0.791 |

Table 6.6 Comparing SBERT-based and GloVe-based *CPP* scores computed between the given source (s) and style-transferred sequences (t).

suggests that SBERT is more sensitive to changes in a text and so applying these contextual-based pre-trained embeddings addresses the issue of the insensitivity of the GloVe-based cosine similarity based *CPP* metrics which has been discussed in some previous (John et al., 2019; Jafaritazehjani et al., 2021, 2022).

Moreover, table 6.2 shows that GloVe-based *CPP* values are quite similar across the domains of sentiment and formality: considering the TST models of $T_{rec}$ and $T_{cyc}$, the average GloVe-based scores for the sentiment and formality domains are respectively 0.975 and 0.9728. However, the SBERT-based values in sentiment domain noticeably differ from those of the formality domain. The average score of the SBERT-based metric for the models $T_{rec}$ and $T_{cyc}$ across the domains of sentiment and formality are 0.6023 and 0.8351, respectively. This can be due to the fact that SBERT model unlike GloVe embedding system captures the negation and opposite sentiment while creating embedding representations of the words. For instance, in sample pair 1 of table 6.6 where source and target sequences have opposite sentiment, i.e. they differ in having and lacking the token *"not"*, SBERT captures the different sentiment of the source and style-transferred texts. Therefore, the cosine similarity score between their embedding vectors is low, 0.26. However, using a GloVe model, the cosine

similarity score of vector representation of the source and target texts of pair 2 is 0.999 which means the two GloVe-based embedding vectors are very similar and the opposite sentiment is not well encoded. In sample pair 2 also SBERT model seems to distinguish between the tokens with different sentiment: *"cool"* and *"cool"* as opposed to *"worse"* and *"worse"*, since it compute a low similarity score of 0.41 for this sequence pair. This does not seem to be the case for GloVe-based metric which assigns a very high score of 0.988 to this pair.

In the formality domain, on the other hand, SBERT-based and GloVe-based *CPP* metrics seem to agree more on the similarity of the source and style-transferred sequence pairs. For instance, pairs 3 and 4 of table 6.6 are sample pairs from formality domain where the reported cosine similarity scores between source and style-transferred embedding vectors created by both SBERT and GloVe models are low (relative to their score ranges).

### 6.4.2 How understanding the encoding of style can inform the design of a TST model

Previous results (section 6.3) suggested that stylistic features of some styles such as formality are more globally encoded. It can be said then that the characteristics of these styles are more implicit and therefore texts having more pervasive features across the text (as opposed to the texts with more locally encoded stylistic features) are more similar to separate languages. If this hypothesis holds, we expect that applying separate decoders for each style and thereby enabling each decoder to specialize in the language distribution of its style should result in a large positive impact on *SSP* for globally encoded styles (such as formality) as compared to more locally encoded styles (such as sentiment). Table 6.7 illustrates that applying style-specific RNN-based decoders leads to an increase of *SSP* and a drop in *CPP* for both style domains compared to the baseline RNN-based model. However, the *SSP* increase is bigger in the formality domain, 17.09% compared to 11.1% in the sentiment domain. The worse

| Dataset | Yelp-*small* (sentiment) | | | GYAFC-$v_2$ (formality) | | |
|---|---|---|---|---|---|---|
| **Model/ Evaluation metrcis** | **CPP** | **PPL** | **SSP** | **CPP** | **PPL** | **SSP** |
| RNN-based baseline model | 0.9261 | **37.98** | 81.8% | 0.9088 | 26.81 | **65.11%** |
| Multi-decoder RNN-based model | 0.9206 | 41.37 | 92.9% | 0.9086 | 31.11 | 82.20% |

Table 6.7 Comparing the results of the single-decoder and multi-decoder RNN-based models in the sentiment and formality domains.

perplexity of the multi-decoder TST models may be due to the fact that less data is available to train each of the decoders.

This observation suggests that applying style-specific decoders is more beneficial to more complicated TST tasks such as formality-shift as opposed to sentiment-shift, i.e. due to characteristics of style domains, some TST architectures can work better in one domain than another style domain. This implies that style characteristics can inform not only the choice of evaluation strategies, but also the selection of TST architectures.

## 6.5 Discussion

The results of the previous chapters revealed that the concept of style is not consistent across the style domains. In this chapter, the focus was firstly on discovering how variations across style domains are encoded. Then, to explore the intersection between domain-specific style characteristics and the TST task. The main findings of the current chapter which address the research questions 3, 4 and 5, respectively (section 1.2). are as follows.

- Sentiment is more locally encoded as compared to the formality which is a more global characteristic of the text. The results also imply that sentiment-shift can be performed by doing less complicated modifications (more token-based) as compared with formality-shift.

  To reach these results, firstly, we examined the effect of putting more emphasis on encoding the input content on the sentiment-shift and formality-shift tasks resulting

in T-based models acting more like an auto-encoder and observed that this resulted in a large drop in *SSP* in the formality domain (section 6.2). Secondly, we conducted a probing classification experiment on the different layers of the transformer encoder which revealed that sentiment characteristics are encoded in lower layers of the encoder as opposed to formality (section 6.3.1). Then, we performed a unigram analysis test comparing the test data and style-shifted sequences which further confirmed the global characteristics of formality compared to sentiment (section 6.3.2).

- Moreover, the results revealed that the variations across the style domains can inform adopting methodologies while dealing with the TST task by showing that GloVe-based metrics are more appropriate in computing the *CPP* of sentiment-shift systems, whereas, SBERT-based *CPP* metrics do a better job in formality domain (section6.4.1).

  We also observed that employing some architectures seems to be more effective in the sentiment domain versus the domain of formality which can be indicative that style characteristics can inform the design of TST architecture (section 6.4.2).

- Finally, the results of the T-based TST models of the current experiments revealed a trade-off between the *CPP* and *SSP* dimensions which mostly holds for the aspect of fluency too[14]. This is inline with the trade-off between the evaluation aspects of the TST task observed in the results of the RNN-based TST systems in the previous chapters. This shows that this trade-off holds while using both RNN-based and transformer-based TST systems which further validates the comprehensive methodology which was proposed and employed throughout the experiments of this research.

[14]T-based baseline model in the domain of formality (table 6.1) has a very low *SSP* score which can mean that it has not converged as a style-shift model and its results have not been considered here.

## 6.6 Conclusion

The findings of this chapter implied that formality is more globally encoded as opposed to sentiment which seems to be a more local characteristic. Studying the variations across style domains is significant since they can contribute to the TST task in different aspects. Firstly, these findings can help provide a more precise definition of the concept of style. Secondly, as the experiments in this chapter showed, domain-specific style characteristics can inform the selection of the methodologies for TST, specifically the *CPP* metric and the selection of TST architectures. We attribute this observation to the specific characteristics of sentiment and formality domains and characteristics of each pre-trained embedding model.

The findings of this research leads us to the future research direction of exploring the interaction between style characteristic and evaluation aspects of *SSP* and fluency. Furthermore, investigating how style characteristics can be taken into account while designing the TST architectures is an interesting direction for future research which can lead us towards improving the TST task more systematically.

| **From negative to positive** |
|---|
| 1. we sit down and we got some really slow and lazy service . |
| 2. we sit down and we got some really quick and efficient service . |
| 3. we sit down and we got some really amazing and sauces service . |
| 4. we sit down and we got some really professional and smile service . |
| 5. we sit guys and we got some really professional and this service . |
| |
| 1. there chips are ok , but their salsa is really bland . |
| 2. there chips are ok , but their salsa is really fresh . |
| 3. there chips are terrific , but their salsa is really flavorful . |
| 4. there chips are perfect , but their salsa is really excellent . |
| 5. there chips are enjoyed , but their salsa is really always . |
| |
| 1. so basically tasted watered down . |
| 2. so basically tasted huge recliners . |
| 3. so basically glad extensive best . |
| 4. so basically tasted delight happy . |
| 5. so basically tasted talented guys . |
| **From positive to negative** |
| 1. the drinks were affordable and a good pour . |
| 2. the drinks were late and a good pour . |
| 3. the drinks were flavorless and a bad pour . |
| 4. the drinks were affordable and a bad pour . |
| 5. the drinks were disappointment and a good pour . |
| |
| 1. my husband got a ruben sandwich , he loved it . |
| 2. my husband got a ruben sandwich , he hated it . |
| 3. my husband got a ruben sandwich , he hated it . |
| 4. my husband got a ruben sandwich , he crap it . |
| 5. my husband got a ruben sandwich , he nothing it . |
| |
| 1. friendly folks , delicious authentic bagels , tasty cream cheese . |
| 2. in nerve , nothing nerve bagels , hard cream cheese . |
| 3. friendly folks , not authentic bagels , tasty cream cheese . |
| 4. rude folks , disappointing authentic bagels , cheap cream cheese . |
| 5. friendly folks , horrible wrong bagels , tasty cream cheese . |

Table 6.8 Style-shifted outputs of the transformer-based TST models using *Yelp-small*. Sequences are ranked as: 1: Input sequence, 2: T-based Baseline model, 3: The proposed T-based model, 4: $T_{rec}$ model, 5: $T_{cyc}$ model.

| **From informal to formal** |
|---|
| 1. be cool and ask her on a date ! |
| 2. be cool and ask her on a date ! |
| 3. be cool and ask her on a date ! |
| 4. be cool and ask her on a date ! |
| 5. be cool and ask her on a date ! |
| |
| 1. ask him to go see a doc . |
| 2. ask him to go see a doc . |
| 3. ask him to go see a doc . |
| 4. ask him to go see a doc . |
| 5. ask him to go see a doc . |
| |
| 1. the show got a little freaky . |
| 2. the show got a little freaky . |
| 3. the show got a little freaky . |
| 4. the show got a little freaky . |
| 5. the show got a little freaky . |
| **From formal to informal** |
| 1. what should i do to be successful with her ? |
| 2. what should i do to be successful with her ? |
| 3. what should i do to be 'd with her ? |
| 4. what should i do to be successful with her ? |
| 5. what should i do to be striptease with her ? |
| |
| 1. i am assuming that you are a republican . |
| 2. i am heave retail heave retail backing heave retail backing |
| 3. i am assuming that you are a republican i am assuming that you are republican |
| 4. i am assuming that you are a republican . |
| 5. i am assuming that you are a republican . |
| |
| 1. you need to do this a few times to get comfortable . |
| 2. you need to do this a few times to get comfortable . |
| 3. you need to do this a few times to get comfortable . |
| 4. you need to do this a few times to get comfortable . |
| 5. you need to do this a few times to get comfortable . |

Table 6.9 Style-shifted outputs of the transformer-based TST models using *GYAFC*-$v_2$. Sequences are ranked as: 1: Input sequence, 2: T-based Baseline model, 3: The proposed T-based model, 4: $T_{rec}$ model, 5: $T_{cyc}$ model.

# Chapter 7

# Summary and future directions

The current chapter summarizes the experimental results, the findings, and the contribution of this research. It also discusses the possible directions for future work.

## 7.1 Summary of the research and main contributions

This thesis has had its main focus on exploring textual style within and across style domains and how it can contribute to the TST task. Specifically, it has investigated the style concept in aspects such as how it is related to content (chapters 4) and its consistency across different style domains (chapter 5). Moreover, in chapter 6, it has examined the style variations across various domains of style and how domain-specific style characteristics and TST problem can interact. This section provides a summary of experimental work, findings and contributions of this research, first, in terms of style characteristics (section 7.1.1) and then in terms of how these characteristics can contribute to the TST task (section 7.1.2). Finally, it reviews the factors that should be considered while evaluating the TST task (section 7.1.3). We discuss the main contributions of the current research in more details here (sections 7.1.1, 7.1.2 and 7.1.3).

### 7.1.1 Investigating style characteristics while framing the TST problem

Throughout experiments of this thesis, several end-to-end adversarial TST systems have been proposed. To explore textual style, firstly, some experiments were designed to study the latent space of these systems. Specifically, a series of probing classifiers were trained to analyse the latent space of several RNN-based adversarial TST models. The results of these experiments (performed in chapters 4 and 5) which indicate the presence of the style of inputs in their corresponding latent representations lead to contribution 1 which directly addresses the research question 1: **Are the two elements of style and content separable?**.

**Contribution 1** indicates that style and content cannot be totally separated and shows that this finding holds across different style domains of sentiment and formality. This observation questions the conceptual basis of the computationally-based strategies used in some previous work on textual style transfer where stylistic features are detected and removed as a preliminary step (Li et al., 2018a; Madaan et al., 2020; Leeftink and Spanakis, 2019; Xu et al., 2018; Zhang et al., 2018a; Sudhakar et al., 2019).

Chapter 5 of this research presented our answer to the reserach question 2: **Is the concept of style consistent across different domains?** Different experiments were conducted to compare the level of entanglement of style and content across sentiment and formality domains. Namely, variational techniques were applied to modify the latent space of the TST frameworks by stripping out the source style features from it. Then the effect of this experiment across different style domains was studied. The experimental results of this chapter revealed that style and content are more entangled in the sentiment domain as opposed to the formality domain. The findings of these experiments led to the **contribution 2** indicating that style is not consistent across different style domains. Many previous research framed the TST problem regardless of stylistic characteristics. This observation questions these approaches which implicitly assume that consistency of the the style concept across different domains.

Considering this observation, chapter 6 proceeded by exploring the research question 3: **How does the encoding of style vary across domains?** Specifically, it applied transformer-based adversarial TST systems to frame the task and designed some probing experiments to study how style features are encoded in different layers of the transformer-encoder of the TST networks. This revealed that sentiment features are more local compared to formality features which are more global. Moreover, other tests including a series of unigram-based experiments further confirmed this observation. These results lead to **contribution 3** stating that the encoding of style can vary.

### 7.1.2 Interaction between style characteristics and TST problem

The findings of this work have indicated that different domains of style are different in terms of how style is encoded. This raised the research question 4: **How do the characteristics of style and the task of TST interact?**

Chapter 6 investigated this interaction by looking into how domain-specific style characteristics can inform the selection of experimental methodologies while dealing with TST problem. In particular, it focused on evaluation methodologies in the case of computing *CPP*. It reported some experiments which indicated that GloVe-based *CPP* metrics work better in computing the *CPP* in the sentiment domain. However, these experiments also revealed that SBERT-based *CPP* metrics do a more precise job in computing the *CPP* while doing formality-shift task. These findings which are inline with the domain-specific characteristics that we observed in these two style domains form **contribution 4** of the current work. This contribution indicates that style characteristics can improve the TST task in different ways such as informing the adoption of evaluation methodology.

### 7.1.3 A comprehensive evaluation methodology for TST problem

TST is a multi-objective task which aims at generating a linguistically fluent text in a desired style while preserving the content of a given text. Which factors to consider while evaluating a TST model is an open question in the field of TST which has been investigated in this thesis as the research question 5: **What factors are relevant for the evaluation of a TST system?** The results of the TST evaluation frameworks used throughout the experiments of the chapters 4, 5 and 6 showed a trade-off between the evaluation dimensions of *SSP*, *CPP* and fluency across different style domains and architectures. This implies that a comprehensive evaluation methodology is needed for the TST task which leads to another contribution of the current work.

**Contribution 5** states that the three dimensions of the TST task should be considered during the evaluation process to form a comprehensive evaluation methodology: content preservation power, style-shift power and fluency. In practice, many state-of-the-art papers on TST including (Fu et al., 2018a; Gröndahl and Asokan, 2020; Hu et al., 2017b; Li et al., 2018b, 2020; Madaan et al., 2020; Xu et al., 2018) do not consider all these three evaluation dimensions. Consequently, papers do not fully validate their approach and hence, they cannot easily be compared. Moreover, the presence of a trade-off between the three aspects of evaluation that has been reported in some previous research (John et al., 2019; Li et al., 2020; Tikhonov et al., 2020) highlights the importance of considering all the three aspects. This trade-off implies that considering one or some aspects and disregarding the other(s) can lead to sacrificing the aspects which were not considered.

## 7.2 Secondary contributions

A variety of other contributions have been made to the TST filed while doing the current research. These secondary contributions are discussed in this section.

1. Our findings clarified style characteristics in various style domains. Doing this from a theoretical perspective, can improve the knowledge of style and help to define this concept more precisely.

2. Clarifying style characteristics from a more practical perspective, can contribute to the TST research by reducing confusion during the manual compositions of parallel data (Dai et al., 2019b). It can also result in creating more clear guidelines for the human evaluation tests in the TST field. This can disambiguate the evaluations process for the human judges which can lead to more reliable human test results with higher inter-annotator agreements.

3. As the results of experiments in section 6.4.2 implied employing multi-decoder strategy improved the formality-shift task better than the sentiment-shift problem. This can be due to the variations between these style domains which can mean that clarifying style characteristics can also be applied to inform the design of TST architectures.

4. The TST task can be framed as a multi-task problem by focusing on the characteristics of the style domain it considers. The idea here is to use the knowledge from one domain to improve the performance of the TST task. Style characteristics can in fact help in applying the appropriate task to frame together with the style-shift problem using multi-task strategies. For instance, Zhang et al. (2020) improved formality-shift (informal to formal) by framing TST together with grammar error correction.

## 7.3 Limitations and future work

This section first describes some limitation and challenges that we faced throughout the course of the current research. It proceeds by discussing some of the possible future directions to extend the current findings and address some of these limitations.

| TST Models | CPP (GloVe-based) | PPL | SSP | Datasets |
|---|---|---|---|---|
| **RNN-based baseline model** | 0.7986 | 7.98 | 78.60% | Newsela-$v_1$ |
| **RNN-based baseline model** | 0.8499 | 15.93 | 66.33% | MSD |
| **RNN-based baseline model** | 0.8015 | 67.36 | 98.38% | Paper-News Title |

Table 7.1 The performance of TST models using datasets of MSD, Paper-News Title and Newsela-$v_1$.

**Limitations** One of the limitations of the current work is that we mainly considered the two styles of sentiment and formality. We tried to broaden the research by including other style domains but the main challenge we faced in doing this was accessing data. We conducted a number of experiments (not reported in the earlier chapters of the thesis) using a few other datasets (other style domains) but our models failed to converge as TST systems. They either had very low *SSP* which can mean that they acted more like auto-encoders or had very low *CPP* and high *SSP* which means they were only able to shift the style without preserving the content. In both cases, it is safe to say that the system is not shifting the textual style successfully. In table 7.1 we report some of these results.

We trained the TST models using the Paper-News Title dataset and also we used the two parallel datasets of MSD and Newsela-$v_1$ in a non-parallel mode. MSD is a parallel simplification corpus which contains 114K textual pairs discussing medical topics and while training it has the vocab-size of 17688. Each sequence pair consist of one professional sample and one simplified version of a medical text (Cao et al., 2020). The Paper-News Title dataset contains 107,538 paper titles crawled from academic websites, and 108,503 news titles in the categories of science and technology category (Fu et al., 2018a) and while training it has the vocab-size of 22180. The style domain of this dataset is topic changing between news and scientific terms. Newsela-$v_1$ (described in section 3.1.3 and which we used for the unigram analysis presented in chapter 6) is a simplification corpus which contains

26K complex sequences and 24K simple sequences (table 3.5) and while training it has the vocab-size of 12088.

Table 7.1 reports the performances of the TST models trained on these datasets where the results show that RNN-based baseline model has very low *CPP* score on the three corpora. These scores are lower or almost the same as the *LB* score of the Glove-based *CPP* scores which is on average 0.85. Given these results, it is unlikely that the proposed T-based model work well if it is trained on these datasets since T-based models are bigger in size compared to RNN-based models. Therefore, due to the time and computational costs, we did not train the proposed T-based model using these corpora. We believe that the small size of these datasets contributed to the fact that the TST systems failed to converge and in future we would like to explore other style domains by using bigger-sized corpora.

**Future directions** This section discusses some of the possible future directions of this research.

- Throughout the experiments of this work, we mostly focused on the style domains of sentiment and formality. Extending this research to other style domains to explore their characteristics is the next step of this study. This focuses on extending the contributions 1 and 2 of this research.

- Contribution 4 of this work states that style characteristics can inform the choice of *CPP* metrics. One future research direction is to extend this finding by studying the interaction of style characteristics and other aspects of evaluation, i.e. *SSP* and fluency.

- Contribution 5 of this report highlights the necessity of considering all the dimensions of the task when evaluating TST systems. However, to the extent of our knowledge, there has not yet been introduced a single evaluation metric for TST representing the overall performance of TST systems. Some previous work such as (Xu et al., 2018; Li et al., 2020) have used geometric mean of CPP and SSP as an single metric. However,

firstly, they did not consider all the three evaluation aspects and, secondly, they did examine the weights and importance of each of the evaluation aspects. As a future direction we would like to proceed towards introducing this single evaluation metric since it can make an important contribution to the field by enabling the ranking and therefore comparing the performance of various TST frameworks.

- One of the future directions can be extending the secondary contribution 3 to further clarify how style characteristics can inform the selection of TST architectures. One potential direction is studying the performance of the multi-decoder transformer-based TST models across different style domains.

- Another potential direction of the future work is to focus on applying the insight of the secondary contribution 4 indicating that style characteristics can inform applying multi-tasking techniques to improve the TST task. For instance, the next step can focus on framing the sentiment-shift task together with POS tagging task.

# Bibliography


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. style transformer. *IEEE transactions on neural networks*, 5(2):157–166.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yixin Cao, Ruihao Shui, Liangming Pan, Min-Yen Kan, Zhiyuan Liu, and Tat-Seng Chua. 2020. Expertise style transfer: A new task towards better communication between experts and laymen. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1061–1071.

Keith Carlson, Allen Riddell, and Daniel N. Rockmore. 2017. Zero-shot style transfer in text using recurrent neural networks. *CoRR*, abs/1711.04731.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of the 28th Neural Information Processing Systems (NIPS), Workshop on Deep Learning*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2067–2075. JMLR.org.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pages 2126–2136.

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. 2019a. Style transformer: Unpaired text style transfer without disentangled latent representation. In *Proceedings*

*of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5997–6007, Florence, Italy. Association for Computational Linguistics.

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. 2019b. Style transformer: Unpaired text style transfer without disentangled latent representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5997–6007, Florence, Italy. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019c. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018a. Style transfer in text: Exploration and evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018b. Style transfer in text: Exploration and evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Proceedings of the 28th conference in Neural Information Processing Systems (NIPS)*, pages 2672–2680. Curran Associates, Inc.

Tommi Gröndahl and N Asokan. 2020. Effective writing style transfer via combinatorial paraphrasing. *Proceedings on Privacy Enhancing Technologies*, 2020(4):175–195.

Sepp Hochreiter and Jürgen Schmidhuber. 1996. Lstm can solve hard long time lag problems. In *Advances in Neural Information Processing Systems*, volume 9. MIT Press.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017a. Controllable text generation. *CoRR*, abs/1703.00955.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017b. Toward controlled generation of text. In *International conference on machine learning*, pages 1587–1596. PMLR.

Philip TG Jackson, Amir Atapour Abarghouei, Stephen Bonner, Toby P Breckon, and Boguslaw Obara. 2019. Style augmentation: data augmentation via style randomization. In *CVPR workshops*, volume 6, pages 10–11.

Somayeh Jafaritazehjani, Gwénolé Lecorvé, Damien Lolive, and John Kelleher. 2020. Style versus content: A distinction without a (learnable) difference? In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2169–2180, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Somayeh Jafaritazehjani, Gwénolé Lecorvé, Damien Lolive, and John D. Kelleher. 2021. Style as sentiment versus style as formality: The same or different? In *ICANN*.

Somayeh Jafaritazehjani, Gwénolé Lecorvé, Damien Lolive, and John D Kelleher. 2022. Local or global: Understanding the variation in the encoding of style across sentiment and formality.

Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. 2022. Deep Learning for Text Style Transfer: A Survey. *Computational Linguistics*, 48(1):155–205.

Zhijing Jin, Di Jin, Jonas Mueller, Nicholas Matthews, and Enrico Santus. 2019. Imat: Unsupervised text attribute transfer via iterative matching and translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3097–3109.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics (TACL)*, 5:339–351.

Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. 2018. Depthwise separable convolutions for neural machine translation. In *International Conference on Learning Representations*.

John D. Kelleher. 2019. *Deep Learning*. MIT Press.

John D. Kelleher, Brian Mac Namee, and Aoife D'arcy. 2020. *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT Press.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

K. Krippendorff. 1980. Content analysis: An introduction to its methodology. Beverly Hills: Sage Publications.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 957–966, Lille, France. PMLR.

Wouter Leeftink and Gerasimos Spanakis. 2019. Towards controlled transformation of sentiment in sentences. In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence-Volume 2: ICAART*, pages 809–816. SCITEPRESS.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018a. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 16th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 1 (Long Papers)*, pages 1865–1874.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018b. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 16th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 1 (Long Papers)*, pages 1865–1874.

Yuan Li, Chunyuan Li, Yizhe Zhang, Xiujun Li, Guoqing Zheng, Lawrence Carin, and Jianfeng Gao. 2020. Complementary auxiliary classifiers for label-conditional text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8303–8310.

Shuming Ma and Xu Sun. 2017. A semantic relevance based neural network for text summarization and text simplification. *Computational Linguistics*, Volume: 1(1).

Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabas Poczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W Black, and Shrimai Prabhumoye. 2020. Politeness transfer: A tag and generate approach. In *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online*, volume 58, pages 1869—-1881,.

David D McDonald and James Pustejovsky. 1985. A computational theory of prose style for natural language generation. In *Second Conference of the European Chapter of the Association for Computational Linguistics*.

Vasudevan Nedumpozhimana and John Kelleher. 2021. Finding BERT's idiomatic key. In *Proceedings of the 17th Workshop on Multiword Expressions (MWE 2021)*, pages 57–62, Online. Association for Computational Linguistics.

Vasudevan Nedumpozhimana, Filip Klubička, and John D. Kelleher. 2022. Shapley idioms: Analysing bert sentence embeddings for general idiom token identification. *Frontiers in Artificial Intelligence*, 5.

Xing Niu, Sudha Rao, and Marine Carpuat. 2018. Multi-task neural models for translating between styles within and across languages. In *COLING*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pages 866–876. Association for Computational Linguistics.

Ella Rabinovich, Raj Nath Patel, Shachar Mirkin, Lucia Specia, and Shuly Wintner. 2017. Personalized machine translation: Preserving original author traits. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Volume 1, Long Papers*, pages 1074–1084. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018a. Improving language understanding by generative pre-training.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018b. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Juan Ramos. 1999. Using tf-idf to determine word relevance in document queries.

Sudha Rao and Joel R Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *NAACL-HLT*.

Sravana Reddy and Kevin Knight. 2016. Obfuscating gender in social media writing. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 17–26.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Alexey Romanov, Anna Rumshisky, Anna Rogers, and David Donahue. 2019. Adversarial decomposition of text representation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 1 (Long and Short Papers)*, pages 815–825.

YOSSI Rubner, CARLO TOMASI, and LEONIDAS J GUIBAS. 2000. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pages 86–96. Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. 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, *Proceedings of the Conference in Neural Information Processing Systems 30 (NIPS)*, pages 6830–6841. Curran Associates, Inc.

Dengliang Shi. 2017. A study on neural network language modeling. *CoRR*, abs/1708.07252.

Prashanth Gurunath Shivakumar and Panayiotis Georgiou. 2019. Confusion2vec: Towards enriching vector space word representations with representational ambiguities. *PeerJ Computer Science*, 5:e195.

Ayush Singh and Ritu Palod. 2018. Sentiment transfer using seq2seq adversarial autoencoders. *CoRR*, abs/1804.04003.

Akhilesh Sudhakar, Bhargav Upadhyay, and Arjun Maheswaran. 2019. "transforming" delete, retrieve, generate approach for controlled text style transfer. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3269–3279.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Conference in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Marc Tanti, Albert Gatt, and Kenneth P Camilleri. 2018. Where to put the image in an image caption generator. *Natural Language Engineering*, 24(3):467.

Youzhi Tian, Zhiting Hu, and Zhou Yu. 2018. Structured content preservation for unsupervised text style transfer. *arXiv preprint arXiv:1810.06526*.

A Tikhonov, V Shibaev, A Nagaev, A Nugmanova, and IP Yamshchikov. 2020. Style transfer for texts: Retrain, report errors, compare with rewrites. In *EMNLP-IJCNLP 2019-2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 3936–3945.

Alexey Tikhonov and Ivan P. Yamshchikov. 2018. What is wrong with style transfer for texts? *CoRR*, abs/1808.04365.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pages 979–988. Association for Computational Linguistics (ACL).

Ruochen Xu, Tao Ge, and Furu Wei. 2019. Formality style transfer with hybrid textual annotations. *arXiv preprint arXiv:1903.06353*.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Ivan Yamshchikov, Viacheslav Shibaev, Nikolay Khlebnikov, and Alexey Tikhonov. 2020. Style-transfer and paraphrase: Looking for a sensible semantic similarity metric. *arXiv preprint arXiv:2004.05001*.

Ivan P Yamshchikov, Viacheslav Shibaev, Aleksander Nagaev, Jürgen Jost, and Alexey Tikhonov. 2019. Decomposing textual information for style transfer. In *3rd Workshop on Neural Generation and Translation, EMNLP-IJCNLP 2019*, pages 128–137. Association for Computational Linguistics.

Yi Zhang, Tao Ge, and Xu Sun. 2020. Parallel data augmentation for formality style transfer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3221–3228.

Yi Zhang, Jingjing Xu, Pengcheng Yang, and Xu Sun. 2018a. Learning sentiment memories for sentiment modification without parallel data. In *EMNLP*.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018b. Style transfer as unsupervised machine translation. *arXiv preprint arXiv:1808.07894*.

Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander Rush, and Yann LeCun. 2018. Adversarially regularized autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5902–5911, Stockholmsmässan, Stockholm Sweden. PMLR.

Xu Zheng, Tejo Chalasani, Koustav Ghosal, Sebastian Lutz, and Aljosa Smolic. 2019. Stada: Style transfer as data augmentation. In *14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2019, Volume 4: VISAPP, Prague, Czech Republic,SciTePress.*, pages 107—-114.

# Appendix A

# Papers

- Style versus Content: A distinction without a (learnable difference)? (Jafaritazehjani et al. 2020). *Proceedings of the 28 th International Conference on Computational Linguistics (Coling 2020), pp.2169-2810, 2020*
- Style as Sentiment versus Style as Formality: the same or different? (Jafaritazehjani et al. 2021). *Proceeding of the 30th International Conference on Artificial Neural Networks (ICANN 2021).*
- Local or Global: Understanding the Variation in the Encoding of StyleAcross Sentiment and Formality (Jafaritazehjani et al. 2022). *under review.*

# Appendix B

# List of Employability and Discipline Specific Skills Training

## B.1 Employability Skills

- RESM 1953 – Research Integrity (5 ECTS)
- SPEC 9160 – Problem Solving, Comm & Creativity (5 ECTS)
- Univ Rennes1 32546 – French Course Intermediate Level- Part 1 (5 ECTS)
- Univ Rennes1 32547 – French Course Intermediate Level- Part 2 (5 ECTS)

## B.2 Discipline Specific Training Skills

- SPEC 9270 – Machine Learning (10 ECTS)
- Univ Bretagne 32582 – Data Analysis and Probabilistic Modelling (5 ECTS)
- DeepLearn 2019, 32584 – 3rd International Summer School on Deep Learning (https://deeplearn2019.irdta.eu) (5 ECTS)