Controlling Neural Language Generation

by

Tianxiao Shen

B.Eng., Tsinghua University (2016)
S.M., Massachusetts Institute of Technology (2018)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author .....
Department of Electrical Engineering and Computer Science
May 13, 2022

Certified by .....
Tommi Jaakkola
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by .....
Regina Barzilay
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Controlling Neural Language Generation

by

Tianxiao Shen

Submitted to the Department of Electrical Engineering and Computer Science
on May 13, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Large-scale neural language models have made impressive strides in natural language generation. However, typical models operate in a left-to-right, unconstrained fashion with limited control over what is generated. This thesis explores flexible sequence models and weakly supervised methods to perform various controlled generation tasks. We anticipate that these techniques will be broadly applicable to other domains, such as the generation of images, molecules, and biological sequences.

We begin by presenting a class of sequence models called *blank language models* (BLMs), which generate sequences by dynamically creating and filling in blanks. Given partially specified text with one or more blanks, BLM will fill in the blanks with a variable number of tokens consistent with the context. Our model is well suited for a variety of text editing and rewriting tasks and demonstrates effectiveness on text infilling, ancient text restoration, and sentiment transfer.

Next, we investigate text autoencoders and their use to control generation through latent space operations. We establish a theory on how to mold a meaningful latent space geometry for discrete text data. Building on this, we develop a family of *denoising text autoencoders* that demonstrate the potential of attribute modification (e.g., tense, sentiment, etc.) through simple vector arithmetic.

The final two chapters address language style transfer in the absence of supervised data. We first formalize the task of non-parallel style transfer and discuss the feasibility of the learning problem. We propose a method that leverages distributional alignment of latent representations to perform style transfer. Then, we study confounding factors and show that by dividing the data into two groups of different styles, with the sets in each group illustrating variations we do not wish to alter, we can exploit invariance to isolate confounders and transfer text in the desired direction.

Thesis Supervisor: Tommi Jaakkola
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Regina Barzilay
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

The six years of working, living and studying at MIT have been an invaluable and unforgettable experience for me. I am grateful for the help and support of many amazing people here and outside, without whom I would not have been able to make this journey.

First and foremost, I would like to express my utmost gratitude to my advisors Regina Barzilay and Tommi Jaakkola. Regina's vision, creativity, and passion, as well as Tommi's brilliance, knowledge, and technical insights are my source of inspiration. Their high standards and encouragement have led to all the exciting research and helped me grow into a scientist. I could not have asked for better advisors.

I would like to thank Yoon Kim for being on my thesis committee and providing sound advice on both research and career. I also owe many thanks to Tao Lei, Yuan Zhang, Jonas Mueller, Marc'Aurelio Ranzato, Karthik Narasimhan, Danqi Chen, and Leslie Kaelbling for their great mentorship and guidance.

I have had the fortune of working with a number of wonderful collaborators and learning from them – thanks to Myle Ott, Michael Auli, Benson Chen, Victor Quach, Hao He, and Yilun Xu. From interesting research discussions to fun activities, my labmates have been a big part of my life at MIT – I especially would like to thank Jiaming Luo, Wengong Jin, Darsh Shah, Adam Yala, Adam Fisch, Yujia Bao, Yujie Qian, Tal Schuster, Rachel Wu, Peter Mikhael, Jeremy Wohlwend, Guang-He Lee, Xiang Fu, Timur Garipov, Octavian Ganea, Bracha Laufer, Tian Xie, Jiang Guo, and Shiyu Chang.

I am also very thankful to my English teachers A.C. Kemp for helping me improve my listening and speaking skills, and Eric Grunwald for reading and writing skills.

I thank the department staff member, especially our graduate administrator Janet Fischer and our administrative assistant Marcia Davidson for their reliability and patience.

I am forever indebted to my parents for their unconditional love and support. I have also been extremely lucky to have wonderful close friends – Peilin Zhong, Wei

Hu, Hao Peng, Zhulin Li, Litian Liu, Jiasi Shen, Siqi Chen, Hengyun Zhou, Ning Jiang, Fan Zhang, who have accompanied and supported me along the way. Finally, I am grateful to Yuqing for his love and care (especially in the quarantine life during the COVID-19 pandemic).

Learning is (and almost has to be) a struggle. But this struggle is also a joyous path, full of wonder and discoveries, a way of life.

– Tommi Jaakkola

The beauty of doing a PhD is that you can find what makes you happy to some degree independently from what others think about it. At the end each one of us has something unique – staying true to oneself ensures that your path in research will be distinct.

– Regina Barzilay

PhD study is (largely) self-supervised learning.

– Tianxiao Shen

Bibliographic Note

Portions of this thesis are based on prior peer-reviewed publications:

- **Chapter 2**: Tianxiao Shen, Victor Quach, Regina Barzilay, Tommi Jaakkola. "Blank language models" [113]
- **Chapter 3**: Tianxiao Shen, Jonas Mueller, Regina Barzilay, Tommi Jaakkola. "Educating text autoencoders: Latent representation guidance via denoising" [112]
- **Chapter 4**: Tianxiao Shen, Tao Lei, Regina Barzilay, Tommi Jaakkola. "Style transfer from non-parallel text by cross-alignment" [110]
- **Chapter 5**: Tianxiao Shen, Regina Barzilay, Tommi Jaakkola. "Text style transfer with confounders" [114]

The code and data for the techniques presented in this thesis are available at `https://github.com/shentianxiao`

Contents

1 Introduction **21**

1.1 Generating Text at Specified Locations 23

1.2 Mapping Discrete Text to Continuous Spaces 24

1.3 Disentangling Different Aspects of Language 26

2 Blank Language Models **29**

2.1 Introduction 29

2.2 Related Work 30

2.3 Blank Language Models 31

2.4 Experiments 36

2.4.1 Text Infilling 37

2.4.2 Ancient Text Restoration 41

2.4.3 Sentiment Transfer 42

2.4.4 Language Modeling 45

2.5 Conclusion 46

3 Denoising Text Autoencoders **49**

3.1 Introduction 49

3.2 Related Work 51

3.3 Methods 52

3.4 Latent Space Geometry 54

3.4.1 A Toy Example 54

3.4.2 Theoretical Analysis 55

3.5 Experiments 58

3.5.1 Neighborhood Preservation 60
3.5.2 Generation-Reconstruction Trade-off 62
3.5.3 Style Transfer via Latent Vector Arithmetic 64
3.5.4 Sentence Interpolation via Latent Space Traversal 67
3.6 Conclusion 68

4 Language Style Transfer from Non-parallel Data 71
4.1 Introduction 71
4.2 Related Work 73
4.3 Formulation 74
4.3.1 Example 1: Gaussian 75
4.3.2 Example 2: Word Substitution 76
4.4 Method 77
4.4.1 Aligned Autoencoder 78
4.4.2 Cross-Aligned Autoencoder 79
4.5 Experimental Setup 81
4.6 Results 84
4.7 Conclusion 87

5 Language Style Transfer with Confounders 89
5.1 Introduction 89
5.2 Related Work 91
5.3 Style Transfer with Confounders 92
5.3.1 Invariant Classifiers 93
5.3.2 The Style Transfer Model 95
5.4 Experiments 97
5.4.1 Sentiment Transfer with Different Punctuation 98
5.4.2 Sentiment Transfer with Different Categories 102
5.5 Conclusion 105

6 Conclusion 107

A Blank Language Models **109**

A.1 Implementation Details for Text Infilling Baselines 109

A.1.1 Insertion Transformer 109

A.1.2 Masked Language Model 109

A.1.3 BERT+LM 110

A.1.4 Seq2seq-full and Seq2seq-fill 110

A.2 Monte-Carlo Estimate of Perplexity 111

A.3 Generation Trajectory 112

B Denoising Text Autoencoders **113**

B.1 Wasserstein Distance 113

B.2 Toy Experiments with Latent Dimension 5 114

B.3 Proofs 115

B.4 Experimental Details 121

B.5 Human Evaluation 122

B.6 Neighborhood Preservation 123

B.7 Generation-Reconstruction Results on the Yahoo Dataset 125

B.8 Additional Examples 126

C Language Style Transfer from Non-parallel Data **129**

C.1 Proof of Lemma 4 129

D Language Style Transfer with Confounders **131**

D.1 Classifier Accuracy in the Real Task 131

D.2 Trade-off of Different Weight Combinations of Loss Terms 132

D.3 Additional Examples 133

List of Figures

2-1 BLM fills in blanks of arbitrary length. 30

2-2 An example trajectory that generates the sentence "customer service is awesome". Each action is a tuple (b, w, l, r), indicating the blank location b selected for expansion, the word w to fill in, whether to create a left blank l, and whether to create a right blank r. 32

2-3 Architecture of the BLM. In the first stage, an index is chosen among all current blank positions. For that location, a word is selected in the second stage. In the final stage, the blank representation is concatenated with the chosen word's embedding and fed into an MLP to determine the creation of the following blanks. 33

2-4 Examples of input and output for text infilling, ancient text restoration, and style transfer tasks. 36

2-5 Example generations of different models for text infilling on Yahoo Answers. Completions are in italic. Invalid completions are in red. For Seq2seq-fill, we present model outputs along with the merged document. 40

2-6 Example generations by BLM for sentiment transfer on Yelp. The first line is the source sentence with masked words in bold. The second line is BLM's completion. The third line is a human reference. 44

3-1 Latent representations learned by AAE and DAAE when mapping clustered sequences in $\mathcal{X} = \{0, 1\}^{50}$ to $\mathcal{Z} = \mathbb{R}^2$. The training data stem from 5 underlying clusters, with 100 sequences sampled from each (colored accordingly by cluster identity). 50

3-2 Illustration of the learned latent geometry by AAE before and after introducing x perturbations. With high-capacity encoder/decoder networks, a standard AAE has no preference over x-z couplings and thus can learn a random mapping between them (Left). Trained with local perturbations $C(x)$, DAAE learns to map similar x to close z to best achieve the denoising objective (Right). 57

3-3 Recall rate of different autoencoders on the Yelp dataset. Quantifying how well the latent geometry preserves text similarity, recall is defined as the fraction of each sentence's 10 nearest neighbors in terms of normalized edit distance whose representations lie among the k nearest neighbors in the latent space. ARAE has a poor recall $< 1\%$ and thus not shown in the plot. 60

3-4 Generation-reconstruction trade-off of various text autoencoders on the Yelp dataset. The "real data" line marks the PPL of a language model trained and evaluated on real data. We strive to approach the lower right corner with both high BLEU and low PPL. The grey box identifies hyperparameters we finalize for respective models. Points of severe collapse (Reverse PPL > 200) are removed from the lower panel. . . . 63

4-1 An overview of the proposed cross-alignment method. $\mathcal{X}_1$ and $\mathcal{X}_2$ are two sentence domains with different styles y_1 and y_2, and $\mathcal{Z}$ is the shared latent content space. Encoder E maps a sentence to its content representation, and generator G generates the sentence back when combining with the original style. When combining with a different style, transferred $\tilde{\mathcal{X}}_1$ is aligned with $\mathcal{X}_2$ and $\tilde{\mathcal{X}}_2$ is aligned with $\mathcal{X}_1$ at the distributional level. 72

4-2 Cross-aligning between x_1 and transferred x_2. For x_1, G is teacher-forced by its words $w_1 w_2 \dots w_t$. For transfered x_2, G is self-fed by previous output logits. The sequence of hidden states $h^0, \dots, h^t$ and $\tilde{h}^0, \dots, \tilde{h}^t$ are passed to discriminator D_1 to be aligned. Note that our first variant aligned autoencoder is a special case of this, where only h^0 and $\tilde{h}^0$, i.e. z_1 and z_2, are aligned. 80

5-1 Different learning scenarios for style transfer. a) With parallel examples, learning of the transfer mapping is supervised. b) With non-parallel, distributionally matched datasets, learning of the transfer mapping is unsupervised. Nevertheless, style is given as the dataset difference, and generation takes place in-distribution. c) With non-parallel, not distributionally matched datasets, style needs to be inferred by excluding confounding differences. Not only is learning of the transfer mapping unsupervised, but generation is also out-of-distribution. 90

5-2 Illustration of the learning of the invariant classifiers. a) The style classifier C_s is trained to be invariant across different pairs of A_i and B_j datasets. b) The orthogonal classifier C_o is trained to highlight changes in sentences other than the style identified by C_s. 94

A-1 Examples of BLM generation trajectory on the Yelp review dataset. . 112

B.2.1 t-SNE visualization of 5-D latent representations learned by AAE and DAAE when mapping clustered sequences in $\mathcal{X} = \{0, 1\}^{50}$ to $\mathcal{Z} = \mathbb{R}^5$. The training data stem from 5 underlying clusters, with 100 sequences sampled from each (colored accordingly by cluster identity). 114

B.6.2 Recall rate of 10 nearest neighbors in the sentence space retrieved by k nearest neighbors in the latent space of different autoencoders on the Yelp and Yahoo datasets. 123

B.7.3 Generation-reconstruction trade-off of various text autoencoders on the Yahoo dataset. The "real data" line marks the PPL of a language model trained and evaluated on real data. We strive to approach the lower right corner with both high BLEU and low PPL. The grey box identifies hyperparameters we finalize for respective models. Points of severe collapse (Reverse PPL $>$ 300) are removed from the right panel. . . . 125

List of Tables

2.1 BLEU scores and perplexity of generated documents by different models for text infilling. The perplexity is measured by a pre-trained left-to-right language model, and the original documents have perplexity 55.8. 39

2.2 Infilling failure rate (%) of seq2seq models. Other methods always produce valid outputs. 39

2.3 CER for ancient text restoration. 42

2.4 Accuracy and BLEU scores for style transfer. 43

2.5 The estimated perplexity of BLM with the number of Monte-Carlo samples on WikiText-103. 45

2.6 Perplexity on the PTB and WikiText datasets. 46

3.1 Examples of 5 nearest neighbors in the latent Euclidean space of AAE and DAAE on the Yelp dataset. 61

3.2 Above: automatic evaluations of vector arithmetic for tense inversion. Below: human evaluation statistics of our model vs. the closest baseline β-VAE. 65

3.3 Examples of vector arithmetic for tense inversion. 65

3.4 Automatic evaluations of vector arithmetic for sentiment transfer. Accuracy (ACC) is measured by a sentiment classifier. The model of Shen et al. [110] is specifically trained for sentiment transfer with labeled data, while our text autoencoders are not. 66

3.5 Examples of vector arithmetic for sentiment transfer. 67

3.6 Interpolations between two input sentences generated by AAE and our model on the Yelp dataset. 68

4.1 Sentiment accuracy of transferred sentences measured by a pretrained classifier. 84

4.2 Human evaluations on sentiment, fluency and overall transfer quality. Fluency rating is from 1 (unreadable) to 4 (perfect). Overall transfer quality is evaluated in a comparative manner, where the judge is shown a source sentence and two transferred sentences, and decides whether they are both good, both bad, or one is better. 84

4.3 Sentiment transfer samples. The first line is an input sentence, the second and third lines are the generated sentences after sentiment transfer by Hu et al. [51] and our cross-aligned autoencoder, respectively. . . . 85

4.4 BLEU scores of word substitution decipherment and word order recovery. 86

5.1 Classifier accuracy in the synthetic task when the spurious correlation is reversed. 100

5.2 Automatic evaluation results of the synthetic sentiment transfer task. Accuracies less than 30 are marked in red. 100

5.3 Example outputs of the synthetic sentiment transfer task. 101

5.4 Automatic evaluation results of sentiment transfer from different categories. 103

5.5 Human evaluation results of sentiment transfer from different categories. 103

5.6 Example outputs of sentiment transfer from different categories. 104

B.6.1 Examples of nearest neighbors in the latent Euclidean space of AAE and DAAE on Yahoo dataset. 124

B.8.2 Additional examples of vector arithmetic for tense inversion. 126

B.8.3 Additional examples of vector arithmetic for sentiment transfer. 127

B.8.4 Interpolations between two input sentences generated by AAE and our model on the Yahoo dataset. 128

D.1.1 Classifier accuracy in the real task when sentiment-category coupling is reversed. 131
D.2.2 Automatic evaluation results of sentiment transfer from different categories. Accuracies less than 30 are marked in red. 132
D.3.3 Additional examples of the synthetic sentiment transfer task. 133
D.3.4 Additional examples of sentiment transfer from different categories. . . 134

Chapter 1

Introduction

Building machines that can generate and communicate in human language is a long-standing goal of artificial intelligence. In recent years, deep learning has brought tremendous progress to language generation. Neural language models can produce realistic text that can be difficult to distinguish from human writing [14]. Neural machine translation is approaching the level of professional human translation and benefiting millions of people to communicate across language barriers [30, 122, 130] Dialogue systems are widely used as virtual assistants for various purposes such as customer service, request routing, or information gathering [27, 141]. Moreover, cross-modal language generation models yield impressive results in applications including speech recognition [6] and image captioning [72].

These advancements largely attribute to architecture development and scaling up. Early neural language models were based on recurrent neural networks (RNNs), which sequentially perform computations along the symbol position [58, 119]. To enhance RNNs, attention mechanisms were subsequently introduced to trace the dependencies between symbols regardless of their distances [7, 78]. After that, the Transformer architecture was proposed, which dispenses with recurrence and completely relies on the attention mechanism to draw global dependencies [122]. Transformer allows for full parallelization and demonstrates surprising scalability, sparking a trend of improving performance by increasing model size and data volume [14, 100, 101].

However, on the other hand, there have been relatively few advances in generative

modeling of language in a flexible and structured way. The traditional left-to-right language model is widely adopted, which generates each word based on the preceding words. While this autoregressive decomposition enables simple and tractable sequence likelihood functions, the resulting rigid generation process makes it difficult to control what is generated and greatly limits the application of the model in many scenarios. For example,

- We may have a beginning and an end of a story that needs to be filled in between; or we may have a draft where certain parts need to be revised; or we may have a template that needs to be filled out in a specific format, such as medical and legal documents. All of these tasks require the model to start from partially specified text – not necessarily at the beginning – and generate the missing fragments.

- We may want to change the tense of a document from present to past, or change the sentiment of a review from negative to positive; we may want to simplify or expand a sentence, or gradually change from one sentence to another and generate smooth and semantically coherent sentence interpolations. If we can map discrete symbolic sequences into a continuous vector space and preserve language structure there, then we could implement text manipulations through vector operations.

- We can express the same meaning in many different ways and may want to rewrite a sentence in a different style, such as switching between informal and formal, between different dialects, or between different personal styles. This requires us to disentangle content and other aspects of language and control them independently.

In general, to achieve fine-grained control over text generation, we need to develop not only new sequence models to explicitly represent the structure of language, but also new methods to cope with the lack of supervised data on many tasks. This dissertation proposes algorithms and techniques to address these challenges. Our

contributions are orthogonal to advances in deep learning, and our methods can generally be implemented with different architectures and scales. Below, we briefly describe the modeling questions we are interested in and how we tackle each of them:

1.1 Generating Text at Specified Locations

To generate at specified locations with a left-to-right language model, one must use sophisticated inference algorithms to find the infilling content that has a high probability along with the surrounding context. Previous work has attempted to solve this highly complex combinatorial problem by making simplifying Markov assumptions and then performing beam search in both forward and backward directions [118], or relaxing discrete words using embedding vectors or soft distributions and then applying gradient backpropagation [73, 99]. These methods are difficult to accommodate multiple infilling segments, need to specify the infilling length in advance, and require high decoding time complexity.

In Chapter 2, we take a different approach and develop a new sequence model, called the *blank language model* (BLM), designed for filling in any blanks in the input text. BLM operates on a dynamic canvas consisting of words and special blank symbols. It iteratively determines which word to place in a blank and whether to split it into new blanks. The model can start from a single blank or partially completed text with blanks at specified locations, and stops generating when no blanks are left to fill. BLM can be efficiently trained using a lower bound of the marginal data likelihood. At test time, we can simply use greedy decoding or beam search algorithms for inference. On the task of filling missing text snippets [146], BLM significantly outperforms all other baselines in terms of both accuracy and fluency. Experiments on style transfer [110] and damaged ancient text restoration [4] demonstrate the potential of this framework for a wide range of applications.

BLM can generate in a fully autoregressive manner by expanding one blank at a time, or partially autoregressive by expanding multiple blanks at the same time. The latter is especially useful in applications that require fast decoding speed, such as

non-autoregressive machine translation [38, 43, 69]. Among the methods proposed for this purpose, the Insertion Transformer [116] and the Levenshtein Transformer [45] are most relevant to our model. Like BLM, both models iteratively modify the entire sequence of incomplete text: the Insertion Transformer also supports dynamic word insertion, but does not allow users to specify where to insert; the Levenshtein Transformer first predicts the number of placeholders in each iteration, and then independently replaces them with tokens. None of them perform as well for text infilling as BLM, which jointly models context and missing content.

Apart from generating text at specified locations, it would also be interesting to compare the language representation learning ability of BLM with other pre-training methods. After randomly masking a portion of tokens in the input text, different training strategies include predicting the masked tokens independently [24], outputting the entire original text [70], and outputting the masked tokens separated by delimiters as a sequence [102]. Our model outperforms these models of the same scale on text infilling. We look forward to the performance of BLM in language generation and representation learning when scaled up.

1.2 Mapping Discrete Text to Continuous Spaces

Sentences are composed of discrete sequences of words with complex structures, and manipulations such as tense and sentiment modification on their surface form are difficult. Text autoencoders offer a continuous approach to manipulating sentences via modifying their latent vector representations. The model consists of an encoder and a decoder network, where the encoder reads a sentence and maps it into a fixed-length vector representation, and the decoder reads the encoded representation from the encoder and reconstructs the original sentence. The overarching goal is to learn latent representations that uncover language structure while smoothly mapping to the data distribution.

A popular class of generative autoencoders is the variational autoencoder (VAE), which uses a probabilistic decoder to approximate the model posterior distribution

and optimize a variational lower bound on the marginal data likelihood [64]. However, when applied to text generation with a powerful autoregressive decoder, VAE suffers from the posterior collapse problem, where the latent representation is ignored and the model degenerates into a conventional language model [12, 19]. To combat posterior collapse, prior work has tried weakening the decoder, such as dropping out its input words [12] or implementing it using a convolutional network with limited window sizes rather than accessing the full history [19, 135]. However, this results in VAEs having worse data likelihood than conventional language models. Other mitigation strategies include annealing the KL divergence term in the VAE objective [12], introducing skip connections between the latent representation and the output layer [25], applying semi-amortized variational inference [62], and adjusting the training dynamics of the encoder and decoder networks [46]. These methods narrowly prevent the KL term from being 0, and the reconstruction loss is still large, which means that the latent representation encodes little information about the sentence.

In Chapter 3, we focus on another class of autoencoders called the adversarial autoencoder (AAE) [81]. It turns an antoencoder into a generative model by using adversarial training to match the aggregated posterior of the latent representations with a given prior distribution [39]. AAE belongs to the more general Wasserstein autoencoder family, which minimizes a penalized form of the Wasserstein distance between the model and the data distributions [121]. Moreover, the AAE objective is related to the VAE objective plus an additional term for maximizing the mutual information between the latent representation and the sentence [144]. AAE ensures that sufficient sentence information is encoded into the latent representation so that it to be used for text manipulation.

Although AAE can encode sufficient sentence information into the latent representation, it still struggles to maintain coherent latent spaces required to perform meaningful text manipulations via latent vector operations. We demonstrate by example that neural encoders do not necessarily map similar sentences to nearby latent vectors. A theoretical explanation for this phenomenon establishes that high-capacity autoencoders can learn an arbitrary mapping between sequences and associated la-

tent representations. To remedy this issue, we augment AAE with a denoising objective [22, 123] where original sentences are reconstructed from perturbed versions (referred to as DAAE). We prove that this simple modification guides the latent space geometry of the resulting model by encouraging the encoder to map similar texts to similar latent representations. In empirical comparisons with various types of autoencoders, our model provides the best trade-off between generation quality and reconstruction capacity. Moreover, the improved geometry of the DAAE latent space enables *zero-shot* text style transfer via simple latent vector arithmetic.

1.3 Disentangling Different Aspects of Language

The goal of controlling sentence style (such as formality, genre, and personal styles) requires us to disentangle what should be altered from orthogonal aspects of sentences that ought to be kept intact. One way to address this disentanglement problem is to provide parallel data, e.g., for formality transfer, to collect corresponding formal expressions for each informal sentence [103]. However, annotating parallel examples is too costly to collect large amounts of data. In many cases, this is even impossible. Suppose we want to transfer between the styles of two authors, there is no parallel data available.

In Chapter 4, we explore style transfer on the basis of non-parallel text, adapting to real-world scenarios. This is an instance of a broad family of problems including machine translation, decipherment, and sentiment modification. We assume a shared latent content distribution across text corpora of different styles, and propose a method that leverages refined alignment of latent representations to perform style transfer. The transferred sentences from one style should match example sentences from the other style as a population. Our cross-alignment method shows effectiveness on three tasks: sentiment modification [52], decipherment of word substitution ciphers [28], and recovery of word order [13].

We demonstrate one of the earliest successes of non-parallel text style transfer, and the problem has been increasingly studied in the field since our work [56]. A

body of work extends our method of optimizing over sequence models and introduces additional techniques such as back-translation to enhance content preservation and language model regularization to improve generation fluency [47, 77, 136]. Another approach is to modify directly in the lexical form by first removing words of one style and then completing the remaining partial sentence with words of another style [71, 113, 129]. Such methods are effective when the style and content words are clearly separated, such as in sentiment transfer. But more often a word has both style and content information, and therefore cannot be simply removed or retained.

In addition to the tasks tested in our experiments, style transfer applications extend to politeness transfer [79], gender style transfer [96], factual/humorous/romantic style transfer [33], political slant transfer [124], debiasing [98], detoxification [106], etc.

To evaluate whether the output successfully transfers the style of the input, we consider the following three criteria: transfer accuracy, content preservation, and fluency. We can automate evaluation by using a style classifier for transfer accuracy, computing BLEU scores or embedding based similarity for content preservation, and using a language model for fluency [32, 66, 71]. However, automatic evaluation metrics have inherent blind spots and need to be combined with human evaluation [71].

In certain tasks, we not only lack parallel examples, but also datasets with a shared content distribution – the available data sources involve additional confounding differences other than the style to be transferred. For example, when transferring between different dialects or between sonnets and tweets, the distribution of speakers or authors differs between styles. In Chapter 5, we look at such cases where we need to infer the desired style from the datasets and remove confounding factors. We show that this inference can be facilitated by dividing the data into two groups whose primary distinction specifies the style to be transferred, and the sets within each group illustrate confounding variations we do not wish to alter. From there, we can first learn an invariant style classifier that removes nuisance variation [3], and then an orthogonal classifier that highlights the confounding cues [133]. The resulting pair of classifiers guide us to transfer text in the specified direction, creating sentences of types not seen during training. Our experiments show that by using positive and

negative review datasets from different categories, we can successfully transfer the sentiment of a sentence without changing its category.

From flexible sequence modeling to rich and structured language representation learning, these pieces of work together represent a step towards controllable text generation. Moreover, our proposed techniques are broadly applicable to generation problems in other domains, especially in discrete domains, such as generating biological sequences and molecules [17, 111].

Chapter 2

Blank Language Models

2.1 Introduction

Neural language models have shown impressive performance across many applications such as machine translation and summarization where the text is generated from scratch [7, 105]. However, a broader set of text generation tasks — including text editing, information fusion, and ancient text restoration — requires the model to start with partially specified text and generate the missing fragments. In the general setup, the input document may have any number of missing spans, and each span may have an unknown number of missing tokens. To perform this text infilling task [146], a model should: (1) provide fine-grained control over the generation location, (2) accommodate a variable number of missing tokens, and (3) respect both the preceding and following context.

Existing approaches focus on adapting left-to-right language models for text infilling. Intricate inference algorithms leveraging dynamic programming or gradient search are proposed to find the filling content that has a high likelihood within the surrounding context [73, 118, 139]. These methods make simplified Markov assumptions, require high decoding time complexity, and cannot adapt to variable infilling length. Alternatively, Donahue et al. [26] predict the concatenation of the infilling content, but do not guarantee that the output will match the number of missing spans in the input.

They also have ____ which ____ .
They also have *ice cream* which *is really good* .

Figure 2-1: BLM fills in blanks of arbitrary length.

In this work, we introduce the Blank Language Model (BLM), which uses a special "_" symbol to control where tokens can be placed. The generation of BLM follows the grammar of replacing a blank with a word and possibly adjoining blanks. By jointly modeling context and missing content, BLM supports the control of generation location and produces consistent infilling of variable length.

Our model can start from a single blank or partial text with blanks in specified locations. It maps the entire input into a sequence of vector representations, and further processes the representations in blank positions to determine the generation action. Generation actions are performed iteratively until there are no blanks. Since multiple trajectories of BLM actions can produce the same final text, we train the model by maximizing a lower bound of the log-likelihood marginalized over trajectories. At test time, we can use simple greedy decoding or beam search to fill in the blanks in the input text.

BLM shows superior performance in text infilling [146], ancient text restoration [4] and style transfer [110], demonstrating its flexibility to generate text in diverse conditions. Our model achieves 92.5% accuracy and BLEU score of 23.1 on the Amazon dataset for sentiment transfer. On the task of restoring ancient text that lost half of the characters, we reduce the error rate by 3.3 points compared to previous methods.

2.2 Related Work

Recent work has explored various sequence models for non-autoregressive machine translation [43]. The Insertion Transformer supports dynamic canvas with word insertion [116], but does not allow users to specify where to insert. The model is unaware of which parts of the canvas are contiguous text spans that should remain intact, and which (potentially scattered) parts need to be filled in. Directly forcing the Insertion Transformer to perform text infilling can therefore lead to suboptimal solutions.

The Levenshtein Transformer combines insertion and deletion through complex policy learning [45]. Its insertion mechanism is a two-stage process in which placeholders are first predicted and then filled-in in a masked language model (MLM) manner. In text infilling where the blanks/placeholders are given, it reduces to an MLM.

MLMs are commonly used in representation learning [24, 57]. To use them in rewriting tasks, one needs to specify the insertion length in advance and heuristically determine the generation order among the masks [31, 37, 125]. Similarly, XL-Net requires absolute positional embedding and thus does not support unknown-length text infilling [115, 134]. BLM provides a natural formulation for generative modeling that can dynamically accommodate insertions of various length.

Another line of work focuses on finding an optimal language generation order, such as syntax-based generation [29] and learning adaptive generation order [44]. These approaches are tailored to generation from scratch in a specific order. Our model instead is attuned for text rewriting, where the missing parts can be located anywhere in the input text, and the algorithm must flexibly complete them.

2.3 Blank Language Models

A blank language model (BLM) generates sequences by creating and filling in blanks. Generation starts with a single blank and ends when there is no blank. In each step, the model selects a blank "__", predicts a word w, and fills the blank with "w", "__ w", "w __", or "__ w __". This way, a blank can be expanded to any number of words.

We define a *canvas* as a sequence of words interspersed with special "__" tokens. The subsequent action is conditioned on this intermediate stage of generation. Suppose the current canvas is $c = (c_1, \ldots, c_n)$ with blanks located at indices $b_1, \ldots, b_k$ (i.e. $c_{b_i} =$ "__", for $i = 1, \ldots, k$). BLM maps this canvas to a distribution over actions specifying how the canvas is to be revised:

$$p(b, w, l, r|c; \theta) = \text{BLM}(c) \tag{2.1}$$

Step t	Canvas c	Action a: Location b	Word w	(Left blank l,	Right blank r)
0.	__#1__	#1	is	Yes	Yes
1.	__#1__ is __#2__	#1	customer	No	Yes
2.	customer __#1__ is __#2__	#2	awesome	No	No
3.	customer __#1__ is awesome	#1	service	No	No
4.	customer service is awesome			-End-	

Figure 2-2: An example trajectory that generates the sentence "customer service is awesome". Each action is a tuple (b, w, l, r), indicating the blank location b selected for expansion, the word w to fill in, whether to create a left blank l, and whether to create a right blank r.

where $b \in \{b_1, \ldots, b_k\}$ is a blank location; w is a word in the vocabulary V; $l, r \in \{0, 1\}$ denote whether or not to create a blank to the left and right of w; and θ are the model parameters. The *action*, defined as the tuple (b, w, l, r) uniquely specifies the next state of canvas (see Fig. 2-2 for illustration).

Alternatively, we can view the actions in BLM as production rules in a grammar. Each blank represents a nonterminal symbol or the start symbol, and the terminal symbols come from the vocabulary V. The production rules are restricted to be of the form "__" $\rightarrow$ "__?w__?" for $w \in V$, where "?" indicates that the preceding symbol is optional. In contrast to context-free grammars, the probability distribution over production rules in BLM is conditioned on the entire canvas generated so far.

Model Architecture We encode the canvas c into a sequence of representations $(z_1, \ldots, z_n)$, and take representations $Z = (z_{b_1}, \ldots, z_{b_k})$ where the blanks are located. Let d denote the dimension of z's. We factorize the joint distribution $p(b, w, l, r|c; \theta)$ into three parts (shown in Fig. 2-3):

1. Choose a blank:

$$p(b_i|c; \theta) = \text{Softmax}(u^T Z) \tag{2.2}$$

where $u \in \mathbb{R}^d$ is a parameter vector to project z's into one-dimensional logits.

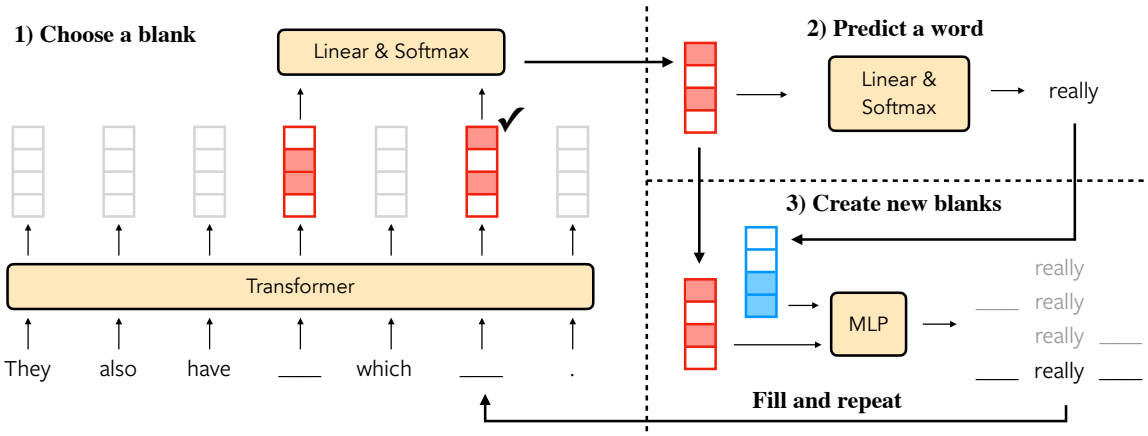

Figure 2-3: Architecture of the BLM. In the first stage, an index is chosen among all current blank positions. For that location, a word is selected in the second stage. In the final stage, the blank representation is concatenated with the chosen word's embedding and fed into an MLP to determine the creation of the following blanks.

2. Predict a word for the selected blank:

$$p(w|c, b_i; \theta) = \text{Softmax}(W z_{b_i}) \tag{2.3}$$

where $W \in \mathbb{R}^{|V| \times d}$ is a parameter matrix to project z_{b_i} into the vocabulary.

3. Decide whether or not to create blanks to the left and right of the predicted word:

$$p(l, r|c, b_i, w; \theta) = \text{MLP}(z_{b_i}, v_w) \tag{2.4}$$

where v_w is the word vector of w, and MLP is a multilayer perceptron with 4 output classes: Left.Yes/No × Right.Yes/No.

Likelihood Now let us consider the probability $p(x; \theta)$ of generating a sentence or paragraph $x = (x_1, \ldots, x_n)$ under the BLM. We call the generating process from an initial blank to complete text a *trajectory*. The same final text x can be realized by multiple trajectories. However, if we specify the order in which the words in x are generated, the trajectory will be uniquely determined. Consider the example trajectory of a 4-word sentence in Fig. 2-2. Given the order $(3, 1, 4, 2)$, at step 0

when we generate x_3, both left and right blanks are created for future generations of x_1 and x_2, x_4. In step 1 of generating x_1, only a right blank is created for the future x_2. Subsequent steps can be deduced by analogy. The correspondence between trajectories and generation orders allows us to write the marginal likelihood as:

$$\begin{aligned} p(x;\theta) &= \sum_{\sigma \in S_n} p(x, \sigma; \theta) \\ &= \sum_{\sigma \in S_n} \prod_{t=0}^{n-1} p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \end{aligned} \tag{2.5}$$

where S_n is the set of all n-permutations; $a_t^{x,\sigma}, c_t^{x,\sigma}$ denote the action and canvas at step t under sentence x and order σ, respectively (cf. Fig. 2-2).

Training Different losses have been proposed to train generalized sequence models. For instance, BERT and XL-Net mask and predict 15% of tokens conditioned on the rest. This strategy is more suitable for representation learning rather than generation. Insertion Transformer masks different numbers of tokens and weights them with uniform loss or binary tree loss [15, 116]. It aims to perform fast inference through parallel decoding. Here, we present a training objective from the language modeling perspective by estimating the log likelihood of generating x.

Directly computing the marginal likelihood over $n!$ orders is intractable. We apply Jensen's inequality to lower bound the log likelihood:

$$\begin{aligned} \log p(x;\theta) &= \log \sum_{\sigma \in S_n} \prod_{t=0}^{n-1} p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \\ &\geq \log(n!) + \frac{1}{n!} \sum_{\sigma \in S_n} \sum_{t=0}^{n-1} \log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \end{aligned} \tag{2.6}$$

where equality holds when the posterior $p(\sigma|x;\theta)$ is uniform. By maximizing this lower bound, we do not favor any particular order, but encourage the model to realize x equally well in all orders. It can help the model to complete any partial input text regardless of the position of blanks.

Algorithm 1 BLM training[1]

Initialize model parameters θ
repeat
  Sample a training example $x = (x_1, \ldots, x_n)$
  Sample t from 0 to $n-1$
  Sample an n-permutation σ
  Construct canvas c that keeps tokens $x_{\sigma_j} (j = 1, \ldots, t)$ and collapses remaining tokens as blanks
  Get $n-t$ target actions a_{j-t} for filling x_{σ_j} $(j = t+1, \ldots, n)$ into canvas c
  Compute loss$(\{a_1, \ldots, a_{n-t}\}, \text{model.forward}(c))$ from Eq. (2.8)
  Update θ by gradient descent
until Convergence

A naive training algorithm is to directly estimate the lower bound in Eq. (2.6): first uniformly sample a permutation σ from S_n and a step t from 0 to $n-1$, then construct the canvas $c_t^{x,\sigma}$, and compute the estimated loss $[-\log(n!) - n \cdot \log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta)]$. However, this procedure has a large variance and can only compute the loss of a single action in one pass (in contrast to left-to-right language models that compute n word losses per pass).

To train the model more efficiently, we note that the canvas $c_t^{x,\sigma}$ depends only on the first t elements of σ. Hence we can combine into one pass the loss calculations of trajectories that are the same in the first t steps but different at the $t+1$ step. Switching the summation order of σ and t, we have:

$$\begin{aligned}
& \sum_{t=0}^{n-1} \frac{1}{n!} \sum_{\sigma \in S_n} \log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \\
= \; & n \cdot \mathbb{E}_t \mathbb{E}_{\sigma_{1:t}} \mathbb{E}_{\sigma_{t+1}} \mathbb{E}_{\sigma_{t+2:n}} \left[\log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \right] \\
= \; & n \cdot \mathbb{E}_t \mathbb{E}_{\sigma_{1:t}} \mathbb{E}_{\sigma_{t+1}} \left[\log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \right] \\
= \; & \mathbb{E}_t \mathbb{E}_{\sigma_{1:t}} \left[\frac{n}{n-t} \sum_{\sigma_{t+1}} \log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \right]
\end{aligned} \tag{2.7}$$

which leads to our efficient training algorithm: sample t from 0 to $n-1$ and partial

[1] We implement a batch version of the algorithm.

They also have ____ which ____ . They also have ice cream which is really good .
τε εγγονον εισαι??????σοφιαι τε εγγονον εισαιου του σοφιαι
The employees were **super nice** and **efficient** ! The employees were rude and unprofessional !

Figure 2-4: Examples of input and output for text infilling, ancient text restoration, and style transfer tasks.

permutation $\sigma_{1:t}$, construct the canvas $c_t^{x,\sigma}$, and compute loss:

$$-\log(n!) - \frac{n}{n-t}\sum_{\sigma_{t+1}} \log p(a_t^{x,\sigma}|c_t^{x,\sigma};\theta) \tag{2.8}$$

The whole process is illustrated in Algorithm 1. In this way, we can compute in expectation $n/2$ action losses per pass.

2.4 Experiments

We test BLM's capacity to rewrite specified portions of text on three tasks: text infilling [146], ancient text restoration [4], and style transfer [110]. Fig. 2-4 displays example inputs and outputs for these tasks. We also measure the perplexity of BLM on language modeling benchmarks and compare with traditional left-to-right language models.

Experimental Details In all experiments, the sequence representations in BLM are obtained using the encoder module of `transformer_base` [122] (6 layers, 8 heads, $d_{model} = 512$, $d_{ff} = 2048$, $d_k = d_v = 64$). The MLP used for blank prediction has one hidden layer of size 1024. Weight decay, learning rate, and dropout are tuned based on the loss on the validation set for each dataset respectively. When decoding, we use beam size in $\{1, 5, 10\}$ and choose the best value as observed on the validation set. We note that beam search in BLM does not search for the sentence with the

maximum marginal likelihood $p(x;\theta)$, but instead for a sentence *and* a trajectory that have the maximum joint likelihood $p(x,\sigma;\theta)$.

2.4.1 Text Infilling

Dataset We experiment on the Yahoo Answers dataset, which has 100K/10K/10K documents for train/valid/test respectively [135]. A document has a maximum length of 200 words, with an average of 78 words. Following Zhu et al. [146], we automatically compile test data by deleting portions of documents. For each document x, we randomly mask a given ratio r of its tokens. Contiguous masked tokens are collapsed into a single "__", resulting in a canvas c to be completed.

Metrics We measure generation's accuracy by computing its BLEU score against the original document x, and fluency as its perplexity evaluated by a pre-trained (left-to-right) language model. We also report the failure rate, which is the percentage of invalid generations, such as missing existing words or not filling in all the blanks.

Baselines We compare BLM with five baselines:

- *Insertion Transformer (InsT)*: By default, InsT does not support controlling the insertion position. We force it to produce valid generations by normalizing the predictions over valid locations, disabling the ⟨eos⟩ prediction unless all blanks have been filled, and prioritizing slots that have not been filled yet. Without these steps, InsT would have a failure rate $\geq 88\%$.
- *MLM (oracle length)*: MLM for text infilling requires predicting the length of each blank. Here we replace blanks with the target number of ⟨mask⟩ tokens, and fill them autoregressively by the most-confident-first heuristic.
- *BERT+LM*: We use BERT's representation of each blank as seed for a left-to-right language model that learns to generate the tokens in the corresponding blank. At inference time, the multiple blanks are filled in one after another, conditioned on previous generations.

- *Seq2seq-full* [26]: We train a seq2seq model to output the full document x from input c. Note that it may have invalid outputs that do not match the input format, such as missing existing tokens in c or generating tokens in incorrect locations.

- *Seq2seq-fill* [26]: We train a seq2seq model to output only tokens to be placed in the blanks, with a special '|' token to indicate separation. For the example in Fig. 2-4, its target output will be "ice cream |is really good". Unlike *seq2seq-full*, *seq2seq-fill* does not have the problem of losing existing tokens in c. However, it may still fail to generate the correct number of '|' that matches the input.

Results As shown in Table 2.1, BLM achieves the highest BLEU score at all mask ratios: 0.7 to 1.7 higher than InsT, 2.6 to 4.1 higher than MLM with oracle length, and 3.7 to 9.4 higher than BERT+LM. InsT is not trained with insertion position control. Restricting it to generate at the specified positions thus biases the model towards making suboptimal completions. MLM is trained to independently predict masked tokens instead of jointly modeling them. Even with the target number of ⟨mask⟩ tokens given, its performance is still inferior to BLM. BERT+LM lags behind other models. In BERT training, one mask corresponds to one token, whereas a blank here can cover multiple tokens, and the distance between words is not fixed. Hence, it is difficult for the LM to complete the sentence from BERT representations.

Seq2seq-full has BLEU scores closest to BLM. However, its failure rate ranges from 15% to 40.6% as the mask ratio increases. Seq2seq-fill performs worse than Seq2seq-full, possibly because the decoder has to model segmented text while counting the number of blanks.

In terms of fluency, outputs of BLM, InsT and Seq2seq-full all have perplexity lower than original data perplexity. This is because with beam search, models tend to generate the most typical output with the highest likelihood [50].

Examination of model generations confirms the superiority of BLM. In Fig. 2-5, we showcase example outputs by each model at different mask ratios. In low mask ratio settings, models only need to fill in the blanks with a single word to produce

Table 2.1: BLEU scores and perplexity of generated documents by different models for text infilling. The perplexity is measured by a pre-trained left-to-right language model, and the original documents have perplexity 55.8.

	BLEU					**PPL**				
Mask ratio	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
No infill	75.2	55.0	37.4	23.6	13.0	98.4	163.0	266.3	421.0	647.9
InsT	84.8	72.3	58.9	46.0	33.8	**48.3**	**44.2**	41.8	39.7	37.7
MLM (oracle length)	83.7	69.3	55.5	43.2	32.2	58.4	59.8	59.8	59.0	56.8
BERT+LM	82.8	66.3	50.3	37.4	26.2	55.1	55.2	54.9	56.5	53.6
Seq2seq-full	86.3	72.9	59.4	46.3	34.0	51.3	46.9	41.0	**31.9**	**20.6**
Seq2seq-fill	82.8	67.5	52.9	39.9	28.6	64.6	71.0	73.4	65.6	48.7
BLM	**86.5**	**73.2**	**59.6**	**46.8**	**34.8**	50.2	44.9	**39.9**	35.0	32.7

Table 2.2: Infilling failure rate (%) of seq2seq models. Other methods always produce valid outputs.

Mask ratio	10%	20%	30%	40%	50%
Seq2seq-full	15.0	22.4	28.7	33.3	40.6
Seq2seq-fill	31.0	28.4	34.5	42.5	47.2

grammatical completions. Most models succeed in this task. With a higher mask ratio of 50%, the main ideas of the document are concealed, and the infilling task is much more challenging. Models need to creatively generate sentences that fit the imposed canvas. Although the original meaning of the sentence is not recovered, BLM is the only model able to produce a coherent document with consistency between the question and the answer.

Overall, BLM displays the best performance both quantitatively and qualitatively. Its inherent text infilling ability frees it from length, order, or termination heuristics used by other methods.

	Mask-ratio 10%
Blanked	when time flies , ____ does it go ? ____ the center of the ____ to be recycled ____ made into new time .
BLM	when time flies , *where* does it go ? *for* the center of the *earth* to be recycled *and* made into new time .
InsT	when time flies , *where* does it go ? *for* the center of the *earth has* to be recycled *and* made into new time .
MLM (oracle len)	when time flies , *where* does it go ? *from* the center of the *earth* to be recycled *converted* made into new time .
BERT+LM	when time flies , *where* does it go ? *to* the center of the *earth* to be recycled *came* made into new time .
Seq2seq-full	when time flies , *where* does it go ? *at* the center of the *earth* to be recycled *and* made into new time .
Seq2seq-fill	when time flies , *how* does it go ? *at* the center of the *earth* to be recycled *and* made into new time . how |at |earth |and
Original	when time flies , **where** does it go ? **to** the center of the **universe** to be recycled **and** made into new time .
	Mask-ratio 50%
Blanked	when time ____ , where ____ ? ____ the ____ of ____ universe to ____ recycled ____ made into ____ .
BLM	when time *was created* , where *did it come from* ? *it was* the *first part* of *the* universe to *be* recycled *and* made into *space* .
InsT	when time *was created* , where *was it* ? *what was* the *name* of *the* universe to *be* recycled *and* made into *space* .
MLM (oracle len)	when time *is* , where *is the universe* ? *from* the *creation* of *the* universe to *be* recycled *and* made into *the universe* .
BERT+LM	when time *is* , where *to* ? *i need to find* the *way* of *the* universe to *be* recycled *and* made into *a lot* .
Seq2seq-full	when time *heals* , where *does it go* ? *it 's* the *end* of *the* universe to *be* recycled *and* made into *space* .
Seq2seq-fill	when time *is time* , where *is time* ? *time is* the *time* of *time* universe to *the* recycled *be* made into *and* . *the universe* is time |is time |time is |time |time |the |be |and |the universe
Original	when time **flies** , where **does it go** ? **to** the **center** of **the** universe to **be** recycled **and** made into **new time** .

Figure 2-5: Example generations of different models for text infilling on Yahoo Answers. Completions are in italic. Invalid completions are in red. For Seq2seq-fill, we present model outputs along with the merged document.

2.4.2 Ancient Text Restoration

Ancient text restoration is a form of text infilling where there are fragments in ancient documents that are illegible due to time-related damages and need to be recovered. Assael et al. [4] introduces the PHI-ML dataset made of fragments of ancient Greek inscriptions. Restoration is performed at the character-level. The number of characters to recover is assumed to be known and indicated by a corresponding number of '?' symbols, as shown in the second row of Fig. 2-4. In reality, when epigraphists restore a deteriorated document, the length of the lost fragment is unknown and needs to be guessed as a first step. While models proposed by Assael et al. [4] relies on expert conjectures, we note that BLM can bypass this limitation and flexibly generate completions without this additional knowledge. However, in order to compute the character error rate (CER) for each '?' and have a fair comparison with previous work, we evaluate our model in the length-aware setting.

Length-aware BLM (L-BLM) We present a variant of BLM adapted to the specific features of this task. The vocabulary V is an alphabet of characters from the ancient Greek language. We extend V with special "__[t]__" tokens that denote the length of the fragment to recover. Specifically, as a preprocessing step, consecutive '?' characters are collapsed into a single "__[t]__" token, where t is the number of '?' symbols. For each such blank token, L-BLM is trained to predict a character to fill in and the length $l \in \{0, \dots, t-1\}$ of the new blank to its left. The length of the new blank on the right is accordingly $t - 1 - l$.

Dataset The PHI-ML dataset contains about 3 million words / 18 million characters. We evaluate models in two settings: *single-slot* and *multi-slot*. For the single-slot setting, we use the testing script of Assael et al. [4] which samples a context of length $L = 1000$ from an inscription, then samples a slot of length $C \in [1, 10]$ from that context. The characters from the slot are replaced with '?' and constitute the target. For the multi-slot setting, we progressively increase the number of slots, yielding mask ratios of 25%, 40% and 50% respectively.

Table 2.3: CER for ancient text restoration.

	Single-slot	**Multi-slot**		
Mask ratio	1%	25%	40%	50%
Human	57.3%	-	-	-
Pythia	32.5%	-	-	-
Pythia-Word	**29.1%**	**36.9%**	42.3%	44.9%
L-BLM	33.7%	37.1%	**37.9%**	**41.6%**

Baselines Assael et al. [4] proposed two models: *Pythia*, a character-level seq2seq-based approach; and *Pythia-Word*, a variant of Pythia that uses both character and word representations as input. During training, the model learns to recover the missing characters of examples where a random slot has been masked. When testing on the multi-slot setting, Pythia(-Word) is applied iteratively with beam size 20 for each slot.

Results Table 2.3 summarizes the CER of all models in both settings. L-BLM achieves similar CER as Pythia in the single-slot setting, significantly outperforming human experts. Augmented with word representations, Pythia-Word further decreases the error rate compared to character-only methods.

In reality, restoring damaged inscriptions requires reconstructing multiple lost fragments. As a larger proportion of text is missing, Pythia-Word's performance is degraded. L-BLM is robust to the setting change and outperforms Pythia-Word at the mask ratio of 40% and 50% by 4.4 and 3.3 points, respectively. We posit that L-BLM's advantage lies in its ability to maximize the joint likelihood of the completions over all slots. In contrast, Pythia-Word's is only aware of one slot at a time, and beam search is performed locally within each slot.

2.4.3 Sentiment Transfer

The goal of sentiment transfer is to modify the sentiment of a sentence while maintaining its topic [110]. An example is described on the third row of Fig. 2-4. Inspired by the way humans perform rewriting, we follow a recent line of work in style transfer

Table 2.4: Accuracy and BLEU scores for style transfer.

	Yelp		Amazon	
	ACC	**BLEU**	**ACC**	**BLEU**
Li et al. [71]	88.7	8.4	48.0	22.8
Zhang et al. [142]	96.6	22.8	84.1	33.9
Wu et al. [128]	91.5	**29.9**	40.2	**41.9**
M&I with MLM	41.5	15.9	31.2	32.1
+ classifier	**97.3**	14.1	75.9	28.5
M&I with BLM	79.6	21.9	52.0	24.7
+ classifier	96.5	21.5	**92.5**	23.1

that adopts a two-step approach [71, 129, 131]:

1. Remove words and expressions of high polarity from the source sentence;
2. Complete the partial sentence with words and expressions of the target sentiment.

Specifically, we adapt the *Mask-And-Infill (M&I)* framework of Wu et al. [129]. We perform Step 1 by training a Bi-LSTM sentiment classifier and masking words whose attention weight is above average. We evaluate the contribution of our model as an infilling module in Step 2 in place of their fine-tuned BERT model. To this end, we train two instances of BLM on the dataset, one for each sentiment. At test time, the corresponding BLM is used to produce completions of the target sentiment.

Wu et al. [129] further train the infilling model with the classifier to improve transfer accuracy. They use soft words relaxation to backprop gradients from the classifier to the generator. For BLM, however, we cannot pick locations or insert blanks as "soft" choices, making it challenging to employ a classifier at training time. Nevertheless, we can easily apply the classifier to guide inference. We sample 10 outputs and keep the one with the highest classifier ranking. It is not slower than beam search with size 10 and can be fully parallelized.

Datasets We test on the Yelp and Amazon review datasets [71, 110]. The Yelp dataset has 450K/4K/1K non-parallel sentences for train/valid/test respectively, and

everyone that i spoke with was **very helpful** and **kind** .
everyone that i spoke with was *rude* and *unprofessional* .
everyone that i spoke with wasn't helpful or kind.

the beans were in the burro in the rice was **nowhere to be** found .
the beans were in the burro in the rice was *the best i* found .
the beans were in the burro and the rice was plentiful

there is **definitely not** enough **room** in that part of the venue .
there is *always* enough *parking* in that part of the venue .
there is so much room in that part of the venue

it is n't **terrible** , but it is **n't** very good either .
it is n't *fancy* , but it is *still* very good either .
it is n't perfect , but it is very good .

Figure 2-6: Example generations by BLM for sentiment transfer on Yelp. The first line is the source sentence with masked words in bold. The second line is BLM's completion. The third line is a human reference.

the Amazon dataset has 555K/2K/1K sentences. Each sentence is labeled as either positive or negative.

Metrics We use evaluation methods introduced by prior work [71, 110]. To assess the accuracy of generated sentences with respect to the target sentiment, we use a pretrained CNN classifier that achieves 97.7% accuracy on the Yelp dataset and 82.2% accuracy on the Amazon dataset. We also measure the BLEU score between transferred sentences and human references.

Results In Table 2.4, we can see that directly applying BLM as the infilling module is significantly better than MLM. The accuracy on Yelp and Amazon datasets is increased by 38.1% and 20.8%, respectively. In addition to the aforementioned problem of MLM being trained to predict masked tokens independently, it must generate the same number of tokens as in the source sentence, whereas our BLM formulation is not subject to this constraint. Our simple use of a classifier at inference time further improves accuracy. It achieves the highest accuracy of 92.5% on Amazon with a small decrease in BLEU, indicating that BLM can easily find high-quality outputs.

In Fig. 2-6, we show examples generated by BLM on Yelp. It can dynamically

Table 2.5: The estimated perplexity of BLM with the number of Monte-Carlo samples on WikiText-103.

#MC samples	1	10	100	1000
Estimated PPL	46.3	44.4	43.3	42.5

adapt to the imposed canvas and fill in blanks with expressions of varied lengths, e.g., "**nowhere to be** found" → "the best i found" and "**definitely not**" → "always". We note that failure cases arise when negative words like "either" are left unmasked; BLM is then unable to produce satisfactory outputs from the canvas.

2.4.4 Language Modeling

Language modeling is a special case of text infilling where sequences are generated from scratch. Traditional left-to-right models dominate this task, but are not suitable for text infilling. Conversely, unconventional sequence models are rarely evaluated on language modeling. Here, we study the perplexity of BLM and Insertion Transformer, and compare them with left-to-right language models to provide additional insights.

We use the Monte-Carlo method to estimate the likelihood in Eq. (2.5) with m samples. While the estimate is unbiased, given that per-word perplexity is a convex function of per-sentence likelihood, sampling estimates like ours are likely yielding a value higher than the actual perplexity (see Appendix A.2 for a proof). As m increases, it converges to the actual perplexity.

Datasets We test on three benchmark datasets: Penn Treebank (PTB) which has about 1M tokens [84], WikiText-2 (WT2) which has 2M tokens, and WikiText-103 (WT103) which has 103M tokens [82].

Results Table 2.5 shows the trend of estimated PPL with the number of samples m. We choose $m = 1000$ in our evaluation, which is close to convergence. Table 2.6 summarizes the perplexity of our model in comparison with previous work. The top results are achieved by the Transformer-XL [23] and the adaptive embed-

Table 2.6: Perplexity on the PTB and WikiText datasets.

	PTB	**WT-2**	**WT-103**
LSTM [41]	82.3	99.3	48.7
TCN [8]	88.7	-	45.2
AWD-LSTM [83]	57.3	**65.8**	-
Transformer [23]	-	-	30.1
Adaptive [5]	-	-	18.7
Transformer-XL [23]	**54.5**	-	**18.3**
InsT (our implementation)	77.3	91.4	39.4
BLM	69.2	81.2	42.5

ding method [5]. They use larger model sizes and supplementary techniques that can also be combined with our model. BLM rivals the Insertion Transformer and outperforms left-to-right language models with LSTM and Temporal Convolutional Network (TCN) architecture. Language modeling seems to still be challenging for free-order models. By reporting the perplexity of unconventional models like BLM, we hope to stimulate future work in this area to close the performance gap with traditional left-to-right models.

2.5 Conclusion

In this paper, we proposed the Blank Language Model for flexible text generation. Given partially specified text with one or more blanks, BLM will fill in the blanks with a variable number of tokens consistent with the context. We demonstrate the effectiveness of our model on various text rewriting tasks, including text infilling, ancient text restoration and style transfer.

The action of BLM consists of selecting a blank and replacing it with a word and possibly adjoining blanks. We train BLM by optimizing a lower bound on the marginal data likelihood that sums over all possible generation trajectories. In this way, we encourage the model to realize a sentence equally well in all orders, which is suitable for filling arbitrary blanks. Appendix A.3 shows examples generated by BLM along with their trajectories. Depending on the application, we could also train the

model to generate in specific orders by placing higher weights on the corresponding trajectories.

BLM has plenty of future applications, including template filling, information fusion, assisting human writing, etc. Moreover, we can extend our formulation to a conditional generative model. Such models can be used in machine translation to support editing and refining translation, as well as in dialogue systems to compose a complete sentence with given elements. While we proposed BLM for language generation, it would also be interesting to compare the representations learned by BLM with those produced by other pre-training methods.

Chapter 3

Denoising Text Autoencoders

3.1 Introduction

Autoencoder-based generative models have become popular tools for advancing controllable text generation such as style or sentiment transfer [12, 52, 110, 143]. By representing sentences as vectors in a latent space, these models offer an attractive continuous approach to manipulating text by means of simple latent vector arithmetic. However, the success of such manipulations rests heavily on the geometry of the latent space representations and its ability to capture underlying sentence semantics. We discover that without additional guidance, fortuitous geometric alignments are unlikely to arise, shedding light on challenges faced by existing methods.

In this work, we focus on the latent space geometry of adversarial autoencoders [81, AAEs]. In contrast to variational autoencoders [64, VAEs], AAEs maintain a strong coupling between their encoder and decoder, ensuring that the decoder does not ignore sentence representations produced by the encoder [12]. The training objective for AAEs consists of two parts: the ability to reconstruct sentences and an additional constraint that the sentence encodings follow a given prior distribution (typically Gaussian). We find that these objectives alone do not suffice to enforce proper latent space geometry: in a toy example with clustered data sequences, a perfectly-trained AAE undesirably mixes different clusters in its latent space (Figure 3-1, Left).

We provide a theoretical explanation for this phenomenon by analyzing high-

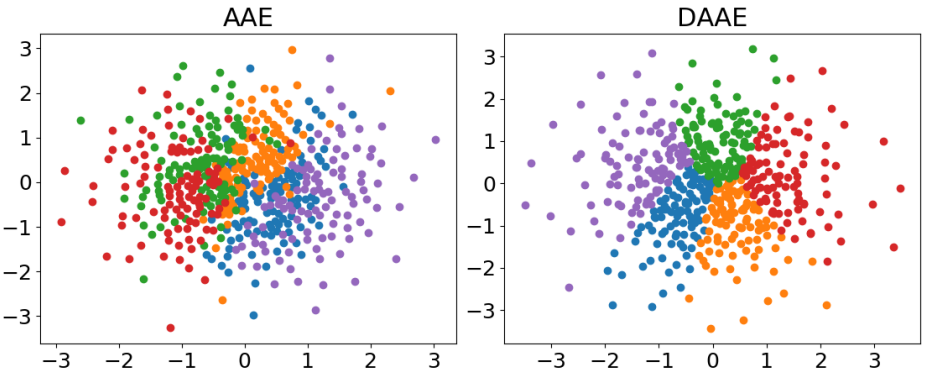

Figure 3-1: Latent representations learned by AAE and DAAE when mapping clustered sequences in $\mathcal{X} = \{0, 1\}^{50}$ to $\mathcal{Z} = \mathbb{R}^2$. The training data stem from 5 underlying clusters, with 100 sequences sampled from each (colored accordingly by cluster identity).

capacity encoder/decoder networks in modern sequence models. For discrete objects such as sentences where continuity assumptions no longer hold, powerful AAEs can learn to map training sentences to latent prior samples arbitrarily, while retaining perfect reconstruction. In such cases, even minimal latent space manipulations can yield random, unpredictable changes in the resulting text.

To remedy this issue, we augment AAEs with a simple denoising objective [22, 123], requiring perturbed sentences (with random words missing) to be reconstructed back to their original versions. We prove that disorganized encoder-decoder mappings are suboptimal under the denoising criterion. As a result, the denoising AAE model (or DAAE for short) will map similar sentences to similar latent representations. Empirical studies confirm that denoising promotes sequence neighborhood preservation, consistent with our theory (Figure 3-1, Right).

Our systematic evaluations demonstrate that DAAE maintains the best trade-off between producing high-quality text vs. informative sentence representations. We further investigate the extent to which text can be manipulated via simple transformations of latent representations. DAAE is able to perform sentence-level vector arithmetic [85] to change the tense or sentiment of a sentence without any supervision during training. Denoising also helps produce higher quality sentence interpolations, suggesting better linguistic continuity in its latent space.

3.2 Related Work

Denoising Vincent et al. [123] first introduced denoising autoencoders (DAEs) to learn robust image representations, and Creswell and Bharath [22] applied DAAEs to generative image modeling. Previous analysis of denoising focused on continuous image data and single-layer networks [95]. Here, we demonstrate that input perturbations are particularly useful for discrete text modeling with powerful sequence networks, as they encourage preservation of data structure in latent space representations.

Variational Autoencoder (VAE) Apart from AAE that this paper focuses on, another popular latent variable generative model is VAE [64]. Unfortunately, when the decoder is a powerful autoregressive model (such as a language model), VAE suffers from the *posterior collapse* problem where the latent representations are ignored [12, 19]. If denoising is used in conjunction with VAE [53] in text applications, then the noisy inputs will only exacerbate VAE's neglect of the latent variable. Bowman et al. [12] proposed to dropout words on the decoder side to alleviate VAE's collapse issue. However, even with a weakened decoder and other techniques including KL-weight annealing and adjusting training dynamics, it is still difficult to inject significant content into the latent code [46, 62, 135]. Alternatives like the β-VAE [48] appear necessary.

Controllable Text Generation Previous work has used autoencoders trained with attribute label information to control text generation [52, 77, 110, 117]. We show that the proposed DAAE can perform text manipulations despite being trained in a completely unsupervised manner without any labels. This suggests that on the one hand, our model can be adapted to semi-supervised learning when a few labels are available. On the other hand, it can be easily scaled up to train one large model on unlabeled corpora and then applied for transferring various styles.

3.3 Methods

Define $\mathcal{X} = \mathcal{V}^m$ as the sentence space of sequences of discrete symbols from vocabulary $\mathcal{V}$ (with length $\leq m$), and let $\mathcal{Z} = \mathbb{R}^d$ denote a continuous space of latent representations. Our goal is to learn a mapping between a data distribution $p_{\text{data}}(x)$ over $\mathcal{X}$ and a given prior distribution $p(z)$ over latent space $\mathcal{Z}$. Such a mapping allows us to manipulate discrete data through its continuous latent representation z, and provides a generative model whereby new data can be sampled by drawing z from the prior and then mapping it to the corresponding sequence in $\mathcal{X}$.

Adversarial Autoencoder (AAE) The AAE involves a deterministic encoder $E : \mathcal{X} \to \mathcal{Z}$ mapping from data space to latent space, a probabilistic decoder $G : \mathcal{Z} \to \mathcal{X}$ that generates sequences from latent representations, and a discriminator $D : \mathcal{Z} \to [0, 1]$ that tries to distinguish between encodings of data $E(x)$ and samples from $p(z)$. Both E and G are recurrent neural nets (RNNs).[1] E takes input sequence x and uses the final RNN hidden state as its encoding z. G generates a sequence x autoregressively, with each step conditioned on z and symbols emitted in preceding steps. D is a feed-forward net that infers the probability of z coming from the prior rather than the encoder. E, G and D are trained jointly with a min-max objective:

$$\min_{E,G} \max_{D} \; \mathcal{L}_{\text{rec}}(\theta_E, \theta_G) - \lambda \mathcal{L}_{\text{adv}}(\theta_E, \theta_D) \tag{3.1}$$

with:

$$\mathcal{L}_{\text{rec}}(\theta_E, \theta_G) = \; \mathbb{E}_{p_{\text{data}}(x)}[-\log p_G(x|E(x))] \tag{3.2}$$

$$\mathcal{L}_{\text{adv}}(\theta_E, \theta_D) = \; \mathbb{E}_{p(z)}[-\log D(z)] + \mathbb{E}_{p_{\text{data}}(x)}[-\log(1 - D(E(x)))] \tag{3.3}$$

[1] We also tried Transformer models [122], but they did not outperform LSTMs on our moderate-size datasets.

where reconstruction loss $\mathcal{L}_{\text{rec}}$ and adversarial loss[2] $\mathcal{L}_{\text{adv}}$ are weighted via hyperparameter $\lambda > 0$ during training.

Denoising Adversarial Autoencoder (DAAE) We extend the AAE by introducing local x-perturbations and requiring reconstruction of each original x from a randomly perturbed version. As we shall see, this implicitly encourages similar sequences to map to similar latent representations, without requiring any additional training objectives. Specifically, given a perturbation process C that stochastically corrupts x to some nearby $\tilde{x} \in \mathcal{X}$, let $p(x, \tilde{x}) = p_{\text{data}}(x)p_C(\tilde{x}|x)$ and $p(\tilde{x}) = \sum_x p(x, \tilde{x})$. The corresponding DAAE training objectives are:

$$\mathcal{L}_{\text{rec}}(\theta_E, \theta_G) = \mathbb{E}_{p(x,\tilde{x})}[-\log p_G(x|E(\tilde{x}))] \tag{3.4}$$

$$\mathcal{L}_{\text{adv}}(\theta_E, \theta_D) = \mathbb{E}_{p(z)}[-\log D(z)] + \mathbb{E}_{p(\tilde{x})}[-\log(1 - D(E(\tilde{x})))] \tag{3.5}$$

Here, $\mathcal{L}_{\text{rec}}$ is the loss of reconstructing x from $\tilde{x}$, and $\mathcal{L}_{\text{adv}}$ is the adversarial loss evaluated using perturbed $\tilde{x}$. This objective simply combines the denoising technique with AAE [22, 123], resulting in the denoising AAE (DAAE) model.

Let $p_E(z|x)$ denote the encoder distribution (for a deterministic encoder it is concentrated at a single point). With perturbation process C, the posterior distributions of the latent representations are of the form:

$$q(z|x) = \sum_{\tilde{x}} p_C(\tilde{x}|x)p_E(z|\tilde{x}) \tag{3.6}$$

This enables the DAAE to utilize stochastic encodings even by merely employing a deterministic encoder network trained without any reparameterization-style tricks. Note that since $q(z|x)$ of the form (3.6) is a subset of all possible conditional distributions, our model is still minimizing an upper bound of the Wasserstein distance

[2]We actually train E to maximize $\log D(E(x))$ instead of $-\log(1 - D(E(x)))$, which is more stable in practice [39]. We also tried the alternative WGAN objective [2] but did not notice any gains.

between data and model distributions, as previously shown by Tolstikhin et al. [121] for AAE (see Appendix B.1 for a full proof).

3.4 Latent Space Geometry

Denoising was previously viewed as a technique to help learn data manifolds and extract more robust representations [10, 123], but there has been little formal analysis of precisely how it helps. Here, we show that denoising guides the latent space geometry of text autoencoders to preserve neighborhood structure in data. By mapping similar text to similar representations, we can perform smoother sentence interpolation and can better implement meaningful text manipulations through latent vector operations.

3.4.1 A Toy Example

We pose the following question: In practice, will autoencoders learn a smooth and regular latent space geometry that reflects the underlying structure of their training data? To study this, we conduct experiments using synthetic data with a clear cluster structure to see if the clusters are reflected in the learned latent representations.

We randomly sample 5 binary (0/1) sequences of length 50 to serve as cluster centers. From each cluster, 100 sequences are sampled by randomly flipping elements of the center sequence with a probability of 0.2. The resulting dataset has 500 sequences from 5 underlying clusters, where sequences stemming from the same cluster typically have many more elements in common than those from different clusters. We train AAE and DAAE models with a latent dimension of 2, so the learned representations can be drawn directly. Similar results were found using a higher latent dimension and visualizing the representations with t-SNE (see Appendix B.2).

In terms of its training objectives, the AAE appears very strong, achieving perfect reconstruction on all data points while keeping the adversarial loss around the maximum $-2 \log 0.5$ (when D always outputs probability 0.5). However, the left panel of Figure 3-1 reveals that, although they are well separated in the data space, different

clusters become mixed together in the learned latent space. This is because that for discrete objects like sequences, neural networks have the capacity to map similar data to distant latent representations. With only the autoencoding and latent prior constraints, the AAE fails to learn proper latent space geometry that preserves the cluster structure of data.

We now train our DAAE model using the same architecture as the AAE, with perturbations C that randomly flip each element of x with probability $p = 0.2$.[3] The DAAE can also keep the adversarial loss close to its maximum, and can perfectly reconstruct the data at test time when C is disabled, indicating that training is not hampered by our perturbations. Moreover, the DAAE latent space closely captures the underlying cluster structure in the data, as depicted in the right panel of Figure 3-1. By simply introducing local perturbations of inputs, we are able to ensure similar sequences have similar representations in the trained autoencoder.

3.4.2 Theoretical Analysis

In this section, we provide theoretical explanations for our previous findings. We formally analyze which type of x-z mappings will be learned by AAE and DAAE, respectively, to achieve global optimality of their training objectives. We show that a well-trained DAAE is guaranteed to learn neighborhood-preserving latent representations, whereas even a perfectly-trained AAE model may learn latent representations whose geometry fails to reflect similarity in the $\mathcal{X}$ space (all proofs are relegated to the appendix).

We study high-capacity encoder/decoder networks with a large number of parameters, as is the case for modern sequence models [24, 101, 107]. Throughout, we assume that:

Assumption 1. *The encoder E is unconstrained and capable of producing any mapping from x's to z's.*

[3] We observed similar results for $p = 0.1, 0.2, 0.3$.

Assumption 2. *The decoder G can approximate arbitrary $p(x|z)$ so long as it remains sufficiently Lipschitz continuous in z. Namely, there exists $L > 0$ such that all decoders G obtainable via training satisfy: $\forall x \in \mathcal{X}, \forall z_1, z_2 \in \mathcal{Z},\ |\log p_G(x|z_1) - \log p_G(x|z_2)| \leq L\|z_1 - z_2\|$. (We denote this set of decoders by $\mathcal{G}_L$.)*

The latter assumption that G is Lipschitz in its continuous input z follows prior analysis of language decoders [87]. When G is implemented as a RNN or Transformer language model, $\log p_G(x|z)$ will remain Lipschitz in z if the recurrent or attention weight matrices have bounded norm, which is naturally encouraged by regularization arising from explicit ℓ_2 penalties and implicit effects of SGD training [140]. Note we have not assumed E or G is Lipschitz in x, which would be unreasonable since x represents discrete text, and when a few symbols change, the decoder likelihood for the entire sequence can vary drastically (e.g., G may assign a much higher probability to a grammatical sentence than an ungrammatical one that only differs by one word).

We further assume an effectively trained discriminator that succeeds in its adversarial task:

Assumption 3. *The discriminator D ensures that the latent encodings $z_1, \ldots, z_n$ of training examples $x_1, \ldots, x_n$ are indistinguishable from prior samples $z \sim p(z)$.*

In all the experiments we have done, training is very stable and the adversarial loss remains around $-2\log 0.5$, indicating that our assumption holds empirically. Under Assumption 3, we can directly suppose that $z_1, \ldots, z_n$ are actual samples from $p(z)$ which are fixed a priori. Here, the task of the encoder E is to map the given training examples to the given latent points, and the goal of the decoder $p_G(\cdot|\cdot)$ is to maximize $-\mathcal{L}_{\text{rec}}$ under the encoder mapping. The question now is which one-to-one mapping between x's and z's an optimal encoder/decoder will learn under the AAE objective (Eq. 3.2) and DAAE objective (Eq. 3.4), respectively.

Theorem 1. *For any one-to-one mapping E from $\{x_1, \ldots, x_n\}$ to $\{z_1, \ldots, z_n\}$, the optimal value of objective $\max_{G \in \mathcal{G}_L} \frac{1}{n} \sum_{i=1}^{n} \log p_G(x_i|E(x_i))$ is the same.*

Intuitively, this result stems from the fact that the model receives no information about the structure of x, and $x_1, \ldots, x_n$ are simply provided as different symbols.

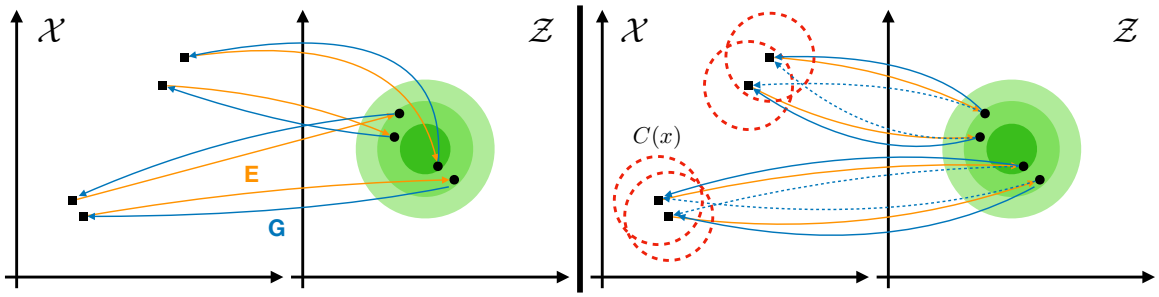

Figure 3-2: Illustration of the learned latent geometry by AAE before and after introducing x perturbations. With high-capacity encoder/decoder networks, a standard AAE has no preference over x-z couplings and thus can learn a random mapping between them (Left). Trained with local perturbations $C(x)$, DAAE learns to map similar x to close z to best achieve the denoising objective (Right).

Hence AAE offers no preference over x-z couplings, and a random matching in which the z do not reflect any data structure is equally good as any other matching (Figure 3-2, Left). Latent point assignments start to differentiate, however, once we introduce local input perturbations.

To elucidate how perturbations affect latent space geometry, it helps to first consider a simple setting with only four examples $x_1, x_2, x_3, x_4 \in \mathcal{X}$. Again, we consider given latent points z_1, z_2, z_3, z_4 sampled from $p(z)$, and the encoder/decoder are tasked with learning which x to match with which z. As depicted in Figure 3-2, suppose there are two pairs of x closer together and also two pairs of z closer together. Let σ denote the sigmoid function, we have the following conclusion:

Theorem 2. *Let d be a distance metric over $\mathcal{X}$. Suppose x_1, x_2, x_3, x_4 satisfy that with some $\epsilon > 0$: $d(x_1, x_2) < \epsilon$, $d(x_3, x_4) < \epsilon$, and $d(x_i, x_j) > \epsilon$ for all other (x_i, x_j) pairs. In addition, z_1, z_2, z_3, z_4 satisfy that with some $0 < \delta < \zeta$: $\|z_1 - z_2\| < \delta$, $\|z_3 - z_4\| < \delta$, and $\|z_i - z_j\| > \zeta$ for all other (z_i, z_j) pairs. Suppose our perturbation process C reflects local $\mathcal{X}$ geometry with: $p_C(x_i|x_j) = 1/2$ if $d(x_i, x_j) < \epsilon$ and $= 0$ otherwise. For $\delta < 1/L \cdot (2\log(\sigma(L\zeta)) + \log 2)$ and $\zeta > 1/L \cdot \log\left(1/(\sqrt{2}-1)\right)$, the denoising objective $\max_{G \in \mathcal{G}_L} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} p_C(x_j|x_i) \log p_G(x_i|E(x_j))$ (where $n = 4$) achieves the largest value when encoder E maps close pairs of x to close pairs of z.*

This entails that DAAE will always prefer to map similar x to similar z. Note that Theorem 1 still applies here, and AAE will not prefer any particular x-z pairing over

the other possibilities. We next generalize beyond the basic four-points scenario to consider n examples of x that are clustered, and ask whether this cluster organization will be reflected in the latent space of DAAE.

Theorem 3. *Suppose $x_1, \ldots, x_n$ are divided into n/K clusters of equal size K, with S_i denoting the cluster index of x_i. Let the perturbation process C be uniform within clusters, i.e. $p_C(x_i|x_j) = 1/K$ if $S_i = S_j$ and $= 0$ otherwise. With a one-to-one encoder mapping E from $\{x_1, \ldots, x_n\}$ to $\{z_1, \ldots, z_n\}$, the denoising objective $\max_{G \in \mathcal{G}_L} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} p_C(x_j|x_i) \log p_G(x_i|E(x_j))$ has an upper bound: $(1/n^2) \cdot \sum_{i,j:S_i \neq S_j} \log \sigma(L\|E(x_i) - E(x_j)\|) - \log K$.*

Theorem 3 provides an upper bound of the DAAE objective that can be achieved by a particular x-z mapping. This achievable limit is substantially better when examples in the same cluster are mapped to the latent space in a manner that is well-separated from encodings of other clusters. In other words, by preserving input space cluster structure in the latent space, DAAE can achieve better objective values and thus is incentivized to learn such encoder/decoder mappings. An analogous corollary can be shown for the case when examples x are perturbed to yield additional inputs $\tilde{x}$ not present in the training data. In this case, the model would aim to collectively map each example and its perturbations to a compact group of z points well-separated from other groups in the latent space.

Our synthetic experiments in Section 3.4.1 confirm that DAAE maintains the cluster structure of sequence data in its latent space. While these are simulated data, we note natural language often exhibits cluster structure based on topics/authorship but also contains far richer syntactic and semantic structures. In the next section, we empirically study the performance of DAAE on real text data.

3.5 Experiments

We test our proposed model and other text autoencoders on two benchmark datasets: *Yelp reviews* and *Yahoo answers* [110, 135]. We analyze their latent space geometry, generation and reconstruction capacities, and applications to controllable text

generation. All models are implemented using the same architecture. Hyperparameters are set to values that produce the best overall generative models (see Section 3.5.2). Detailed descriptions of training settings, human evaluations, and additional results/examples can be found in appendix.

Datasets The Yelp dataset is from Shen et al. [110], which has 444K/63K/127K sentences of less than 16 words in length as train/dev/test sets, with a vocabulary of 10K. It was originally divided into positive and negative sentences for style transfer between them. Here we discard the sentiment label and let the model learn from all sentences indiscriminately. Our second dataset of Yahoo answers is from Yang et al. [135]. It was originally document-level. We perform sentence segmentation and keep sentences with length from 2 to 50 words. The resulting dataset has 495K/49K/50K sentences for train/dev/test sets, with vocabulary size 20K.

Perturbation Process We randomly delete each word with probability p, so that perturbations of sentences with more words in common will have a larger overlap. We also tried replacing each word with a <mask> token or a random word from the vocabulary. We found that all variants have similar generation-reconstruction trade-off curves; in terms of neighborhood preservation, they are all better than other autoencoders, but word deletion has the highest recall rate. This may be because word replacement cannot perturb sentences of different lengths to each other even if they are similar. Defining sentence similarity and meaningful perturbations are task specific. Here, we demonstrate that even the simplest word deletions can bring significant improvements. We leave it to future work to explore more sophisticated text perturbations.

Baselines We compare our proposed DAAE with four alternative text autoencoders: AAE [81], latent-noising AAE [104, LAAE], adversarially regularized autoencoder [143, ARAE], and β-VAE [48]. Similar to our model, the LAAE uses Gaussian perturbations in the latent space (rather than perturbations in the sentence space) to improve AAE's latent geometry. However, it requires enforcing an L_1

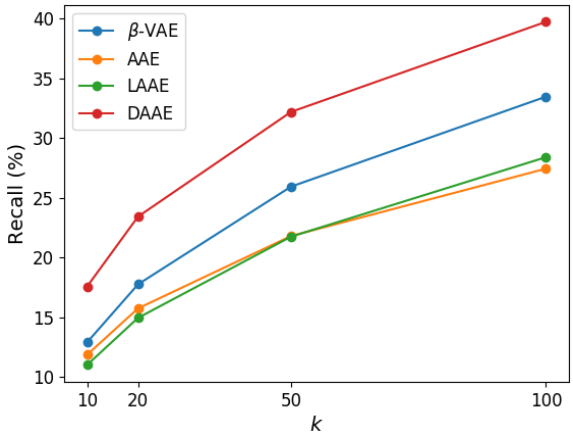

Figure 3-3: Recall rate of different autoencoders on the Yelp dataset. Quantifying how well the latent geometry preserves text similarity, recall is defined as the fraction of each sentence's 10 nearest neighbors in terms of normalized edit distance whose representations lie among the k nearest neighbors in the latent space. ARAE has a poor recall $< 1\%$ and thus not shown in the plot.

penalty $(\lambda_1 \cdot \| \log \sigma^2(x) \|_1)$ on the latent perturbations' log-variance to prevent them from vanishing. In contrast, input perturbations in DAAE enable stochastic latent representations without parametric restrictions like Gaussianity.

3.5.1 Neighborhood Preservation

We begin by investigating whether input perturbations will induce latent space organization that better preserves neighborhood structure in the sentence space. Under our word-dropout perturbation process, sentences with more words in common are more likely to be perturbed into one another. This choice of C approximately encodes sentence similarity via normalized edit distance.[4] Thus, within the test set, we find both the 10 nearest neighbors of each sentence based on the normalized edit distance (denote this set by NN_x), as well as the k nearest neighbors based on Euclidean distance between latent representations (denote this set by NN_z). We compute the recall rate $|\text{NN}_x \cap \text{NN}_z| \ / \ |\text{NN}_x|$, which indicates how well local neighborhoods are preserved in the latent space of different models.

[4]Normalized edit distance $\in [0, 1]$ is the Levenshtein distance divided by the max length of two sentences.

Table 3.1: Examples of 5 nearest neighbors in the latent Euclidean space of AAE and DAAE on the Yelp dataset.

Source	**my waitress katie was fantastic , attentive and personable .**
AAE	my cashier did not smile , barely said hello .
	the service is fantastic , the food is great .
	the employees are extremely nice and helpful .
	our server kaitlyn was also very attentive and pleasant .
	the crab po boy was also bland and forgettable .
DAAE	the manager , linda , was very very attentive and personable .
	stylist brenda was very friendly , attentive and professional .
	the manager was also super nice and personable .
	my server alicia was so sweet and attentive .
	our waitress ms. taylor was amazing and very knowledgeable .
Source	**i have been known to eat two meals a day here .**
AAE	i have eaten here for _num_ years and never had a bad meal ever .
	i love this joint .
	i have no desire to ever have it again .
	you do n't need to have every possible dish on the menu .
	i love this arena .
DAAE	you can seriously eat one meal a day here .
	i was really pleased with our experience here .
	ive been coming here for years and always have a good experience .
	i have gone to this place for happy hour for years .
	we had _num_ ayce dinner buffets for _num_ on a tuesday night .

Figure 3-3 shows that DAAE consistently gives the highest recall, about 1.5∼2 times that of AAE, implying that input perturbations have a substantial effect on shaping the latent space geometry. Table 3.1 presents the five nearest neighbors found by AAE and DAAE in their latent space for example test set sentences. The AAE sometimes encodes entirely unrelated sentences close together, while the latent space geometry of the DAAE is structured based on key words such as "attentive" and "personable", and tends to group sentences with similar semantics close together. These findings are consistent with our previous conclusions in Section 3.4.

3.5.2 Generation-Reconstruction Trade-off

In this section, we evaluate various generative autoencoders in terms of both generation quality and reconstruction accuracy. A strong model should not only generate high quality sentences, but also learn useful latent representations that capture significant data content. Recent work on text autoencoders has found an inherent tension between these aims [12], yet only when both goals are met can we successfully manipulate sentences by modifying their latent representation (in order to produce valid output sentences that retain the semantics of the input).

We compute the BLEU score [93] between input and reconstructed sentences to measure reconstruction accuracy, and compute Forward/Reverse PPL to measure sentence generation quality [21, 143].[5] Forward PPL is the perplexity of a language model trained on real data and evaluated on generated data. It measures the fluency of the generated text, but cannot detect the collapsed case where the model repeatedly generates a few common sentences. Reverse PPL is the perplexity of a language model trained on generated data and evaluated on real data. It takes into account both the fluency and diversity of the generated text. If a model generates only a few common sentences, a language model trained on it will exhibit poor PPL on real data.

We thoroughly investigate the performance of different models and the trade-off between generation and reconstruction. Figure 3-4 plots reconstruction BLEU (higher is better) vs. Forward/Reverse PPL (lower is better). The lower right corner indicates an ideal situation where good reconstruction accuracy and generation quality are both achieved. For models with tunable hyperparameters, we sweep the full spectrum of their generation-reconstruction trade-off by varying the KL coefficient β of β-VAE, the log-variance L_1 penalty λ_1 of LAAE, and the word drop probability p of DAAE.[6]

In the upper panel, we observe that a standard VAE ($\beta = 1$) completely collapses and ignores the latent variable z, resulting in a reconstruction BLEU close to 0. At the

[5] While some use importance sampling estimates of data likelihood to evaluate VAEs [46], adopting the encoder as a proposal density is not suited for AAE variants, as they are optimized based on Wasserstein distances rather than likelihoods and lack closed-form posteriors.

[6] We also studied the VAE with word dropout on the decoder side proposed by Bowman et al. [12], but found that it exhibited poor reconstruction over all settings of the dropout parameter (best BLEU = 12.8 with dropout rate = 0.7). Thus this model is omitted from our other analyses.

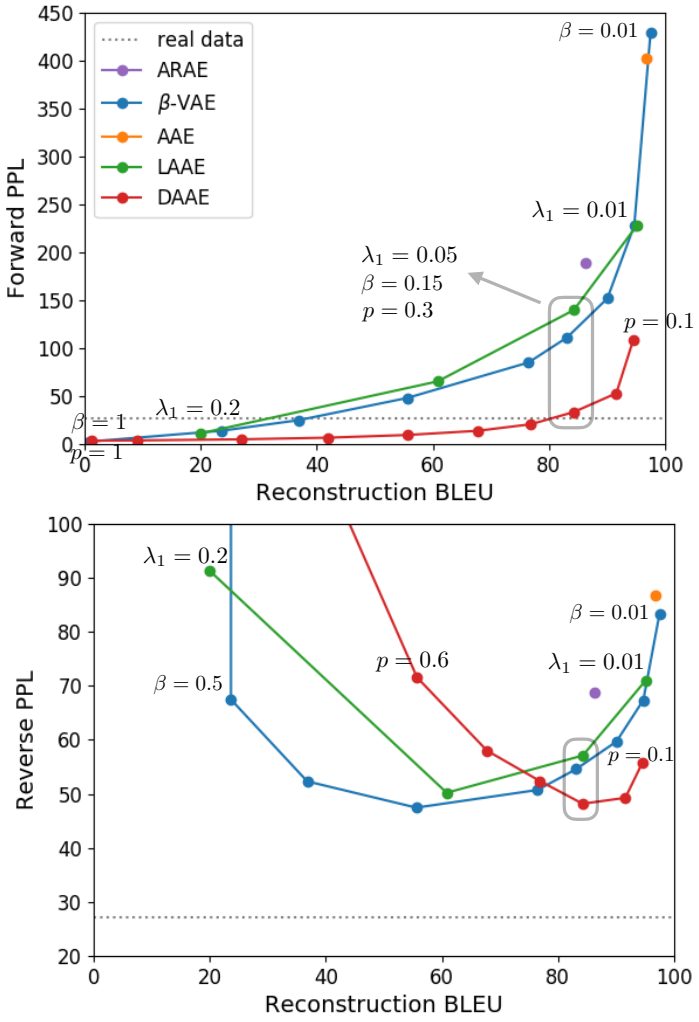

Figure 3-4: Generation-reconstruction trade-off of various text autoencoders on the Yelp dataset. The "real data" line marks the PPL of a language model trained and evaluated on real data. We strive to approach the lower right corner with both high BLEU and low PPL. The grey box identifies hyperparameters we finalize for respective models. Points of severe collapse (Reverse PPL > 200) are removed from the lower panel.

other extreme, AAE can achieve near-perfect reconstruction, but its latent space is highly non-smooth and generated sentences are of poor quality, indicated by its large Forward PPL. Decreasing β in VAE or introducing latent noises in AAE provides the model with a similar trade-off curve between reconstruction and generation. We note that ARAE falls on or above their curves, revealing that it does not fare better than these methods (Cífka et al. [21] also reported similar findings). Our proposed DAAE provides a trade-off curve that is strictly superior to other models. With discrete x and a complex encoder, the Gaussian perturbations added to the latent space by β-VAE and LAAE are not directly related to how the inputs are encoded. In contrast, input perturbations added by DAAE can constrain the encoder to maintain coherence between neighboring inputs in an end-to-end fashion and help learn smoother latent space.

The lower panel in Figure 3-4 illustrates that Reverse PPL first drops and then rises as we increase the degree of regularization/perturbation. This is because when z encodes little information, generations from prior-sampled z lack enough diversity to cover the real data. Again, DAAE outperforms other models that tend to have higher PPL and lower BLEU.

Based on these results, we set $\beta = 0.15$ for β-VAE, $\lambda_1 = 0.05$ for LAAE, and $p = 0.3$ for DAAE in the neighborhood preservation and text manipulation experiments, to ensure they have strong reconstruction abilities and encode enough information about data.

3.5.3 Style Transfer via Latent Vector Arithmetic

Mikolov et al. [85] previously discovered that word embeddings from unsupervised learning can capture linguistic relationships via simple arithmetic. A canonical example is the embedding arithmetic "King" - "Man" + "Woman" $\approx$ "Queen". Here, we use the Yelp dataset with tense and sentiment as two example attributes [52, 110] to investigate whether analogous structure emerges in the latent space of our sentence-level models.

Table 3.2: Above: automatic evaluations of vector arithmetic for tense inversion. Below: human evaluation statistics of our model vs. the closest baseline β-VAE.

Model	ACC	BLEU	PPL
ARAE	17.2	55.7	59.1
β-VAE	49.0	43.5	44.4
AAE	9.7	**82.2**	37.4
LAAE	43.6	37.5	55.8
DAAE	**50.3**	54.3	**32.0**

β-VAE is better: 25	DAAE is better: **48**	
both good: 26	both bad: 67	n/a: 34

Table 3.3: Examples of vector arithmetic for tense inversion.

Input	**i enjoy hanging out in their hookah lounge .**
ARAE	i enjoy hanging out in their 25th lounge .
β-VAE	i made up out in the backyard springs salad .
AAE	i enjoy hanging out in their brooklyn lounge .
LAAE	i enjoy hanging out in the customized and play .
DAAE	i enjoyed hanging out in their hookah lounge .
Input	**had they informed me of the charge i would n't have waited .**
ARAE	amazing egg of the may i actually !
β-VAE	had they help me of the charge i would n't have waited .
AAE	have they informed me of the charge i would n't have waited .
LAAE	they are girl (the number so i would n't be forever .
DAAE	they have informed me of the charge i have n't waited .

Tense We use the Stanford Parser to extract the main verb of a sentence and determine the sentence tense based on its part-of-speech tag. We compute a single "tense vector" by averaging the latent code z separately for 100 (non-parallel) past tense sentences and present tense sentences in the development set, and then calculating the difference between the two. Given a sentence from the test set, we attempt to change its tense from past to present or from present to past through simple addition/subtraction of the tense vector. More precisely, a source sentence x is first is encoded to $z = E(x)$, and then the tense-modified sentence is produced via $G(z \pm v)$, where $v \in \mathbb{R}^d$ denotes the fixed tense vector.

To quantitatively compare different models, we compute their tense transfer ac-

Table 3.4: Automatic evaluations of vector arithmetic for sentiment transfer. Accuracy (ACC) is measured by a sentiment classifier. The model of Shen et al. [110] is specifically trained for sentiment transfer with labeled data, while our text autoencoders are not.

Model		**ACC**	**BLEU**	**PPL**
Shen et al. [110]		81.7	12.4	38.4
AAE	$\pm v$	7.2	86.0	33.7
	$\pm 1.5v$	25.1	59.6	59.5
	$\pm 2v$	57.5	27.4	139.8
DAAE	$\pm v$	36.2	40.9	40.0
	$\pm 1.5v$	73.6	18.2	54.1
	$\pm 2v$	91.8	7.3	61.8

curacy as measured by the parser, output BLEU with the input sentence, and output (forward) PPL evaluated by a language model. DAAE achieves the highest accuracy, lowest PPL, and relatively high BLEU (Table 3.2, Above), indicating that the output sentences produced by our model are more likely to be of high quality and of the proper tense, while remaining similar to the source sentence. A human evaluation on 200 test sentences (100 past and 100 present, details in Appendix B.5) suggests that DAAE outperforms β-VAE twice as often as it is outperformed, and it successfully inverts tense for $(48 + 26)/(200 - 34) = 44.6\%$ of sentences, 13.8% more than β-VAE (Table 3.2, Below). Table 3.3 shows the results of adding or subtracting this fixed latent vector offset under different models. DAAE properly changes "enjoy" to "enjoyed" or the subjunctive mood to declarative mood. Other baselines either fail to alter the tense, or undesirably change the semantic meaning of the source sentence (e.g. "enjoy" to "made").

Sentiment Following the same procedure to alter tense, we compute a "sentiment vector" v from 100 negative and positive sentences and use it to change the sentiment of test sentences. Table 3.4 reports automatic evaluations, and Table 3.5 shows examples generated by AAE and DAAE. Scaling $\pm v$ to $\pm 1.5v$ and $\pm 2v$, we find that resulting sentences get more and more positive/negative. However, the PPL for AAE increases rapidly with the scaling factor, indicating that the sentences become un-

Table 3.5: Examples of vector arithmetic for sentiment transfer.

Input		**the food is entirely tasteless and slimy .**
AAE	$+v$	the food is entirely tasteless and slimy .
	$+1.5v$	the food is entirely tasteless and slimy .
	$+2v$	the food is entirely and beef .
DAAE	$+v$	the food is tremendous and fresh .
	$+1.5v$	the food is sensational and fresh .
	$+2v$	the food is gigantic .
Input		**i really love the authentic food and will come back again .**
AAE	$-v$	i really love the authentic food and will come back again .
	$-1.5v$	i really but the authentic food and will come back again .
	$-2v$	i really but the worst food but will never come back again .
DAAE	$-v$	i really love the authentic food and will never come back again .
	$-1.5v$	i really do not like the food and will never come back again .
	$-2v$	i really did not believe the pretentious service and will never go back .

natural when their encodings have a large offset. DAAE enjoys a much smoother latent space than AAE. At this challenging *zero-shot* setting where no style labels are provided during training, DAAE with $\pm 1.5v$ is able to transfer sentiment fairly well.

3.5.4 Sentence Interpolation via Latent Space Traversal

We also study sentence interpolation by traversing the latent space of text autoencoders. Given two input sentences, we encode them to z_1, z_2 and decode from $tz_1 + (1-t)z_2$ $(0 \leq t \leq 1)$. Ideally, this should produce fluent sentences with gradual semantic change. Table 3.6 shows two examples from the Yelp dataset, where it is clear that DAAE produces more coherent and natural interpolations than AAE. Table B.8.4 in the appendix shows two difficult examples from the Yahoo dataset, where we interpolate between dissimilar sentences. While it is challenging to generate semantically correct sentences in these cases, the latent space of our model exhibits continuity on topic and syntactic structure.

Table 3.6: Interpolations between two input sentences generated by AAE and our model on the Yelp dataset.

Input 1	**it 's so much better than the other chinese food places in this area .**
Input 2	**better than other places .**
AAE	it 's so much better than the other chinese food places in this area .
	it 's so much better than the other food places in this area .
	better , much better .
	better than other places .
	better than other places .
DAAE	it 's so much better than the other chinese food places in this area .
	it 's much better than the other chinese places in this area .
	better than the other chinese places in this area .
	better than the other places in charlotte .
	better than other places .
Input 1	**fried dumplings are a must .**
Input 2	**the fried dumplings are a must if you ever visit this place .**
AAE	fried dumplings are a must .
	fried dumplings are a must .
	the dumplings are a must if you worst .
	the fried dumplings are a must if you ever this place .
	the fried dumplings are a must if you ever visit this place .
DAAE	fried dumplings are a must .
	fried dumplings are a must visit .
	fried dumplings are a must in this place .
	the fried dumplings are a must we ever visit this .
	the fried dumplings are a must if we ever visit this place .

3.6 Conclusion

This paper provided a thorough analysis of the latent space representations of text autoencoders. We showed that simply minimizing the divergence between data and model distributions cannot ensure that the data structure is preserved in the latent space, but straightforward denoising techniques can greatly improve text representations. We offered a theoretical explanation for these phenomena by analyzing the latent space geometry arisen from input perturbations. Our results may also help explain the success of BERT [24], whose masked language modeling objective is similar to a denoising autoencoder.

Our proposed DAAE substantially outperforms other text autoencoders in both generation and reconstruction capabilities, and demonstrates the potential for various text manipulations via simple latent vector arithmetic. Future work may explore more sophisticated perturbation strategies besides the basic random word deletion, or investigate what additional properties of latent space geometry help provide finer control over text generation with autoencoders. Beyond our theory which considered autoencoders that have perfectly optimized their objectives, we hope to see additional analyses in this area that account for the initialization/learning-process and analyze other types of autoencoders.

Chapter 4

Language Style Transfer from Non-parallel Data

4.1 Introduction

Using massive amounts of parallel data has been essential for recent advances in text generation tasks, such as machine translation and summarization. However, in many text generation problems, we can only assume access to non-parallel or mono-lingual data. Problems such as decipherment or style transfer are all instances of this family of tasks. In all of these problems, we must preserve the content of the source sentence but render the sentence consistent with desired presentation constraints (e.g., style, plaintext/ciphertext).

The goal of controlling one aspect of a sentence such as style independently of its content requires that we can disentangle the two. However, these aspects interact in subtle ways in natural language sentences, and we can succeed in this task only approximately even in the case of parallel data. Our task is more challenging here. We merely assume access to two corpora of sentences with the same distribution of content albeit rendered in different styles. Our goal is to demonstrate that this distributional equivalence of content, if exploited carefully, suffices for us to learn to map a sentence in one style to a style-independent content vector and then decode it to a sentence with the same content but a different style.

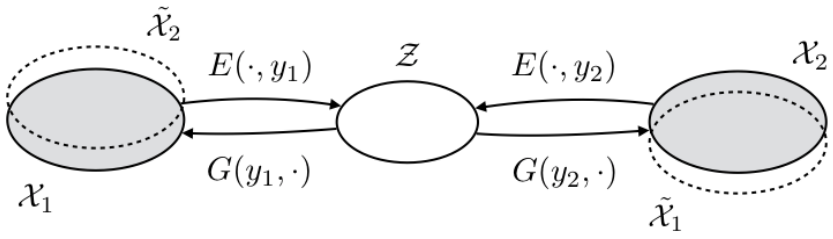

Figure 4-1: An overview of the proposed cross-alignment method. $\mathcal{X}_1$ and $\mathcal{X}_2$ are two sentence domains with different styles y_1 and y_2, and $\mathcal{Z}$ is the shared latent content space. Encoder E maps a sentence to its content representation, and generator G generates the sentence back when combining with the original style. When combining with a different style, transferred $\tilde{\mathcal{X}}_1$ is aligned with $\mathcal{X}_2$ and $\tilde{\mathcal{X}}_2$ is aligned with $\mathcal{X}_1$ at the distributional level.

In this paper, we introduce a refined alignment of sentence representations across text corpora (illustrated in Figure 4-1). We learn an encoder that takes a sentence and its original style indicator as input, and maps it to a style-independent content representation. This is then passed to a style-dependent decoder for rendering. We do not use typical VAEs for this mapping since it is imperative to keep the latent content representation rich and unperturbed. Indeed, richer latent content representations are much harder to align across the corpora and therefore they offer more informative content constraints. Moreover, we reap additional information from cross-generated (style-transferred) sentences, thereby getting two distributional alignment constraints. For example, positive sentences that are style-transferred into negative sentences should match, as a population, the given set of negative sentences.

To demonstrate the flexibility of the proposed model, we evaluate it on three tasks: sentiment modification, decipherment of word substitution ciphers, and recovery of word order. In all of these applications, the model is trained on non-parallel data. On the sentiment modification task, the model successfully transfers the sentiment while keeps the content for 41.5% of review sentences according to human evaluation, compared to 41.0% achieved by the control-gen model of [51]. It achieves strong performance on the decipherment and word order recovery tasks, reaching BLEU score of 57.4 and 26.1 respectively, obtaining 50.2 and 20.9 gap than a comparable method without cross-alignment.

4.2 Related Work

Style Transfer in Vision Non-parallel style transfer has been extensively studied in computer vision [36, 59, 74, 75, 120, 137, 145]. Gatys et al. [36] explicitly extract content and style features, and then synthesize a new image by combining "content" features of one image with "style" features from another. More recent approaches learn generative networks directly via generative adversarial training [39] from two given data domains X_1 and X_2. The key computational challenge in this non-parallel setting is aligning the two domains. For example, CoupledGANs [74] employ weight-sharing between networks to learn cross-domain representation, whereas CycleGAN [145] introduces cycle consistency which relies on transitivity to regularize the transfer functions. While our approach has a similar high-level architecture, the discreteness of natural language does not allow us to reuse these models and necessitates the development of new methods.

Non-parallel Transfer in Natural Language In natural language processing, most tasks that involve generation (e.g., translation and summarization) are trained using parallel sentences. Our work most closely relates to approaches that do not utilize parallel data, but instead guide sentence generation from an indirect training signal [51, 88]. For instance, Mueller et al. [88] manipulate the hidden representation to generate sentences that satisfy a desired property (e.g., sentiment) as measured by a corresponding classifier. However, their model does not necessarily enforce content preservation. More similar to our work, Hu et al. [51] aims at generating sentences with controllable attributes by learning disentangled latent representations [18]. Their model builds on variational autoencoders (VAEs) and uses independency constraints to enforce that attributes can be reliably inferred back from generated sentences. While our model builds on distributional cross-alignment for the purpose of style transfer and content preservation, these constraints can be added in the same way.

Adversarial Training over Discrete Samples Recently, a wide range of techniques addresses challenges associated with adversarial training over discrete sam-

ples generated by recurrent networks [16, 49, 68, 138]. In our work, we employ the Professor-Forcing algorithm [68] which was originally proposed to close the gap between teacher-forcing during training and self-feeding during testing for recurrent networks. This design fits well with our scenario of style transfer that calls for cross-alignment. By using continuous relaxation to approximate the discrete sampling process [55, 80], the training procedure can be effectively optimized through back-propagation [40, 67].

4.3 Formulation

In this section, we formalize the task of non-parallel style transfer and discuss the feasibility of the learning problem. We assume the data are generated by the following process:

1. a latent style variable y is generated from some distribution $p(y)$;
2. a latent content variable z is generated from some distribution $p(z)$;
3. a datapoint x is generated from conditional distribution $p(x|y,z)$.

We observe two datasets with the same content distribution but different styles y_1 and y_2, where y_1 and y_2 are unknown. Specifically, the two observed datasets $X_1 = \{x_1^{(1)}, \ldots, x_1^{(n)}\}$ and $X_2 = \{x_2^{(1)}, \ldots, x_2^{(m)}\}$ consist of samples drawn from $p(x_1|y_1)$ and $p(x_2|y_2)$ respectively. We want to estimate the style transfer functions between them, namely $p(x_1|x_2; y_1, y_2)$ and $p(x_2|x_1; y_1, y_2)$.

A question we must address is when this estimation problem is feasible. Essentially, we only observe the marginal distributions of x_1 and x_2, yet we are going to recover their joint distribution:

$$p(x_1, x_2|y_1, y_2) = \int_z p(z)p(x_1|y_1, z)p(x_2|y_2, z)dz \tag{4.1}$$

As we only observe $p(x_1|y_1)$ and $p(x_2|y_2)$, y_1 and y_2 are unknown to us. If two different y and y' lead to the same distribution $p(x|y) = p(x|y')$, then given a dataset

X sampled from it, its underlying style can be either y or y'. Consider the following two cases: (1) both datasets X_1 and X_2 are sampled from the same style y; (2) X_1 and X_2 are sampled from style y and y' respectively. These two scenarios have different joint distributions, but the observed marginal distributions are the same. To prevent such confusion, we constrain the underlying distributions as stated in the following proposition:

Proposition 1. *In the generative framework above, x_1 and x_2's joint distribution can be recovered from their marginals only if for any different $y, y' \in \mathcal{Y}$, distributions $p(x|y)$ and $p(x|y')$ are different.*

This proposition basically says that X generated from different styles should be "distinct" enough, otherwise the transfer task between styles is not well defined. While this seems trivial, it may not hold even for simplified data distributions. The following examples illustrate how the transfer (and recovery) becomes feasible or infeasible under different model assumptions. As we shall see, for a certain family of styles $\mathcal{Y}$, the more complex distribution for z, the more probable it is to recover the transfer function and the easier it is to search for the transfer.

4.3.1 Example 1: Gaussian

Consider the common choice that $z \sim \mathcal{N}(0, I)$ has a centered isotropic Gaussian distribution. Suppose a style $y = (A, b)$ is an affine transformation, i.e. $x = Az+b+\epsilon$, where ϵ is a noise variable. For $b = 0$ and any orthogonal matrix A, $Az + b \sim N(0, I)$ and hence x has the same distribution for any such styles $y = (A, 0)$. In this case, the effect of rotation cannot be recovered.

Interestingly, if z has **a more complex distribution**, such as a Gaussian mixture, then affine transformations can be uniquely determined.

Lemma 4. *Let z be a mixture of Gaussians $p(z) = \sum_{k=1}^{K} \pi_k \mathcal{N}(z; \mu_k, \Sigma_k)$. Assume $K \geq 2$, and there are two different $\Sigma_i \neq \Sigma_j$. Let $\mathcal{Y} = \{(A, b)||A| \neq 0\}$ be all invertible affine transformations, and $p(x|y, z) = \mathcal{N}(x; Az + b, \epsilon^2 I)$, in which ϵ is a noise. Then for all $y \neq y' \in \mathcal{Y}$, $p(x|y)$ and $p(x|y')$ are different distributions.*

Theorem 5. *If the distribution of z is a mixture of Gaussians which has more than two different components, and x_1, x_2 are two affine transformations of z, then the transfer between them can be recovered given their respective marginals.*

4.3.2 Example 2: Word Substitution

Consider here another example when z is a bi-gram language model and a style y is a vocabulary in use that maps each "content word" onto its surface form (lexical form). If we observe two realizations x_1 and x_2 of the same language z, the transfer and recovery problem becomes inferring a word alignment between x_1 and x_2.

Note that this is a simplified version of language decipherment or translation. Nevertheless, the recovery problem is still sufficiently hard. To see this, let $M_1, M_2 \in \mathcal{R}^{n \times n}$ be the estimated bi-gram probability matrix of data X_1 and X_2 respectively. Seeking the word alignment is equivalent to finding a permutation matrix P such that $P^\top M_1 P \approx M_2$, which can be expressed as an optimization problem,

$$\min_P \ \|P^\top M_1 P - M_2\|^2$$

The same formulation applies to graph isomorphism (GI) problems given M_1 and M_2 as the adjacency matrices of two graphs, suggesting that determining the existence and uniqueness of P is at least GI hard. Fortunately, if M as a graph is complex enough, the search problem could be more tractable. For instance, if each vertex's weights of incident edges as a set is unique, then finding the isomorphism can be done by simply matching the sets of edges. This assumption largely applies to our scenario where z is a complex language model. We empirically demonstrate this in the results section.

The above examples suggest that z as the latent content variable should carry most complexity of data x, while y as the latent style variable should have relatively simple effects. We construct the model accordingly in the next section.

4.4 Method

Learning the style transfer function under our generative assumption is essentially learning the conditional distribution $p(x_1|x_2; y_1, y_2)$ and $p(x_2|x_1; y_1, y_2)$. Unlike in vision where images are continuous and hence the transfer functions can be learned and optimized directly, the discreteness of language requires us to operate through the latent space. Since x_1 and x_2 are conditionally independent given the latent content variable z,

$$\begin{aligned} p(x_1|x_2; y_1, y_2) &= \int_z p(x_1, z|x_2; y_1, y_2)dz \\ &= \int_z p(z|x_2, y_2) \cdot p(x_1|y_1, z)dz \\ &= \mathbb{E}_{z\sim p(z|x_2,y_2)}[p(x_1|y_1, z)] \end{aligned} \tag{4.2}$$

This suggests us learning an autoencoder model. Specifically, a style transfer from x_2 to x_1 involves two steps—an encoding step that infers x_2's content $z \sim p(z|x_2, y_2)$, and a decoding step which generates the transferred counterpart from $p(x_1|y_1, z)$. In this work, we approximate and train $p(z|x, y)$ and $p(x|y, z)$ using neural networks (where $y \in \{y_1, y_2\}$).

Let $E : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be an encoder that infers the content z for a given sentence x and a style y, and $G : \mathcal{Y} \times \mathcal{Z} \to \mathcal{X}$ be a generator that generates a sentence x from a given style y and content z. E and G form an autoencoder when applying to the same style, and thus we have reconstruction loss,

$$\begin{aligned} \mathcal{L}_{\text{rec}}(\theta_E, \theta_G) = \ & \mathbb{E}_{x_1\sim X_1}[-\log p_G(x_1|y_1, E(x_1, y_1))] + \\ & \mathbb{E}_{x_2\sim X_2}[-\log p_G(x_2|y_2, E(x_2, y_2))] \end{aligned} \tag{4.3}$$

where θ are the parameters to estimate.

In order to make a meaningful transfer by flipping the style, X_1 and X_2's content space must coincide, as our generative framework presumed. To constrain that x_1 and x_2 are generated from the same latent content distribution $p(z)$, one option is to apply a variational autoencoder [64]. A VAE imposes a prior density $p(z)$, such as

$z \sim \mathcal{N}(0, I)$, and uses a KL-divergence regularizer to align both posteriors $p_E(z|x_1, y_1)$ and $p_E(z|x_2, y_2)$ to it,

$$\mathcal{L}_{\text{KL}}(\theta_E) = \mathbb{E}_{x_1 \sim X_1}[D_{\text{KL}}(p_E(z|x_1, y_1) \| p(z))] + \mathbb{E}_{x_2 \sim X_2}[D_{\text{KL}}(p_E(z|x_2, y_2) \| p(z))] \quad (4.4)$$

The overall objective is to minimize $\mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{KL}}$, whose opposite is the variational lower bound of data likelihood.

However, as we have argued in the previous section, restricting z to a simple and even distribution and pushing most complexity to the decoder may not be a good strategy for non-parallel style transfer. In contrast, a standard autoencoder simply minimizes the reconstruction error, encouraging z to carry as much information about x as possible. On the other hand, it lowers the entropy in $p(x|y, z)$, which helps to produce meaningful style transfer in practice as we flip between y_1 and y_2. Without explicitly modeling $p(z)$, it is still possible to force distributional alignment of $p(z|y_1)$ and $p(z|y_2)$. To this end, we introduce two constrained variants of autoencoder.

4.4.1 Aligned Autoencoder

Dispense with VAEs that make an explicit assumption about $p(z)$ and align both posteriors to it, we align $p_E(z|y_1)$ and $p_E(z|y_2)$ with each other, which leads to the following constrained optimization problem:

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \mathcal{L}_{\text{rec}}(\theta_E, \theta_G) \\ \text{s.t.} \quad E(x_1, y_1) &\overset{\text{d}}{=} E(x_2, y_2) \qquad x_1 \sim X_1, x_2 \sim X_2 \end{aligned} \quad (4.5)$$

In practice, a Lagrangian relaxation of the primal problem is instead optimized. We introduce an adversarial discriminator D to align the aggregated posterior distribution of z from different styles [81]. D aims to distinguish between these two distributions:

$$\mathcal{L}_{\text{adv}}(\theta_E, \theta_D) = \mathbb{E}_{x_1 \sim X_1}[-\log D(E(x_1, y_1))] + \mathbb{E}_{x_2 \sim X_2}[-\log(1 - D(E(x_2, y_2)))] \quad (4.6)$$

The overall training objective is a min-max game played among the encoder E, generator G and discriminator D. They constitute an aligned autoencoder:

$$\min_{E,G} \max_{D} \mathcal{L}_{\text{rec}} - \lambda \mathcal{L}_{\text{adv}} \tag{4.7}$$

We implement the encoder E and generator G using single-layer RNNs with GRU cell. E takes an input sentence x with initial hidden state y, and outputs the last hidden state z as its content representation. G generates a sentence x conditioned on latent state (y, z). To align the distributions of $z_1 = E(x_1, y_1)$ and $z_2 = E(x_2, y_2)$, the discriminator D is a feed-forward network with a single hidden layer and a sigmoid output layer.

4.4.2 Cross-Aligned Autoencoder

The second variant, cross-aligned autoencoder, directly aligns the transfered samples from one style with the true samples from the other. Under the generative assumption, $p(x_2|y_2) = \int_{x_1} p(x_2|x_1; y_1, y_2)p(x_1|y_1)dx_1$, thus x_2 (sampled from the left-hand side) should exhibit the same distribution as transferred x_1 (sampled from the right-hand side), and vice versa. Similar to our first model, the second model uses two discriminators D_1 and D_2 to align the populations. D_1's job is to distinguish between real x_1 and transferred x_2, and D_2's job is to distinguish between real x_2 and transferred x_1.

Adversarial training over the discrete samples generated by G hinders gradients propagation. Although sampling-based gradient estimator such as REINFORCE [126] can by adopted, training with these methods can be unstable due to the high variance of the sampled gradient. Instead, we employ two recent techniques to approximate the discrete training [51, 68]. First, instead of feeding a single sampled word as the input to the generator RNN, we use the softmax distribution over words instead. Specifically, during the generating process of transferred x_2 from $G(y_1, z_2)$, suppose at time step t the output logit vector is v_t. We feed its peaked distribution $\text{softmax}(v_t/\gamma)$ as the next input, where $\gamma \in (0, 1)$ is a temperature parameter.

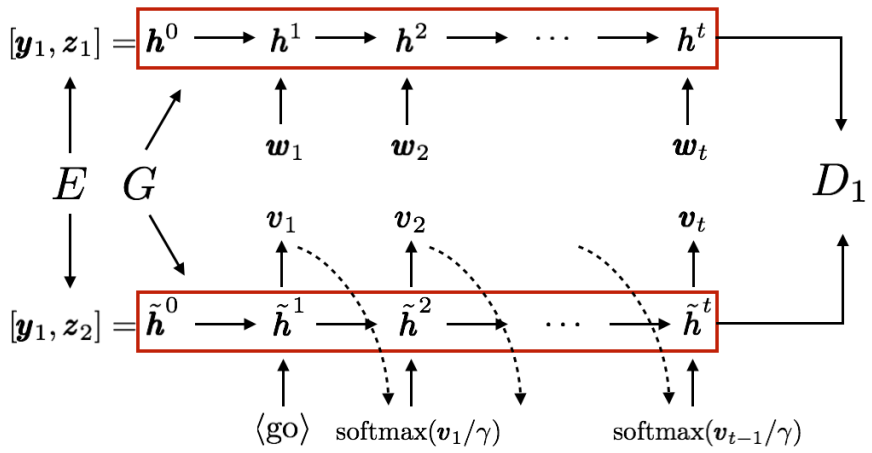

Figure 4-2: Cross-aligning between x_1 and transferred x_2. For x_1, G is teacher-forced by its words $w_1 w_2 \dots w_t$. For transfered x_2, G is self-fed by previous output logits. The sequence of hidden states $h^0, \dots, h^t$ and $\tilde{h}^0, \dots, \tilde{h}^t$ are passed to discriminator D_1 to be aligned. Note that our first variant aligned autoencoder is a special case of this, where only h^0 and $\tilde{h}^0$, i.e. z_1 and z_2, are aligned.

Secondly, we use Professor-Forcing [68] to match the sequence of hidden states instead of the output words, which contains the information about outputs and is smoothly distributed. That is, the input to the discriminator D_1 is the sequence of hidden states of either (1) $G(y_1, z_1)$ teacher-forced by a real example x_1, or (2) $G(y_1, z_2)$ self-fed by previous soft distributions.

The running procedure of our cross-aligned autoencoder is illustrated in Figure 4-2. Note that cross-aligning strengthens the alignment of latent variable z over the recurrent network of generator G. By aligning the whole sequence of hidden states, it prevents z_1 and z_2's initial misalignment from propagating through the recurrent generating process, as a result of which the transferred sentence may end up somewhere far from the target domain.

We implement both D_1 and D_2 using convolutional neural networks for sequence classification [61]. The training algorithm is presented in Algorithm 2.

Algorithm 2 Cross-aligned autoencoder training. The hyper-parameters are set as $\lambda = 1, \gamma = 0.001$ and learning rate is 0.0001 for all experiments in this paper.

Require: Two corpora of different styles X_1, X_2. Lagrange multiplier λ, temperature γ.

Initialize $\theta_E, \theta_G, \theta_{D_1}, \theta_{D_2}$

repeat

for $p = 1, 2$; $q = 2, 1$ **do**

Sample a mini-batch of k examples $\{x_p^{(i)}\}_{i=1}^k$ from X_p

Get the latent content representations $z_p^{(i)} = E(x_p^{(i)}, y_p)$

Unroll G from initial state $(y_p, z_p^{(i)})$ by feeding $x_p^{(i)}$, and get the hidden states sequence $h_p^{(i)}$

Unroll G from initial state $(y_q, z_p^{(i)})$ by feeding previous soft output distribution with temperature γ, and get the transferred hidden states sequence $\tilde{h}_p^{(i)}$

end for

Compute the reconstruction $\mathcal{L}_{\text{rec}}$ by Eq. (4.3)

Compute D_1's (and symmetrically D_2's) loss:

$$\mathcal{L}_{\text{adv}_1} = -\frac{1}{k}\sum_{i=1}^{k} \log D_1(h_1^{(i)}) - \frac{1}{k}\sum_{i=1}^{k} \log(1 - D_1(\tilde{h}_2^{(i)})) \tag{4.8}$$

Update $\{\theta_E, \theta_G\}$ by gradient descent on loss

$$\mathcal{L}_{\text{rec}} - \lambda(\mathcal{L}_{\text{adv}_1} + \mathcal{L}_{\text{adv}_2}) \tag{4.9}$$

Update θ_{D_1} and θ_{D_2} by gradient descent on loss $\mathcal{L}_{\text{adv}_1}$ and $\mathcal{L}_{\text{adv}_2}$ respectively

until convergence

Ensure: Style transfer functions $G(y_2, E(\cdot, y_1)) : \mathcal{X}_1 \to \mathcal{X}_2$ and $G(y_1, E(\cdot, y_2)) : \mathcal{X}_2 \to \mathcal{X}_1$

4.5 Experimental Setup

Sentiment Modification Our first experiment focuses on text rewriting with the goal of changing the underlying sentiment, which can be regarded as "style transfer" between negative and positive sentences. We run experiments on Yelp restaurant reviews, utilizing readily available user ratings associated with each review. Following standard practice, reviews with rating above three are considered positive, and those below three are considered negative. While our model operates at the sentence level, the sentiment annotations in our dataset are provided at the document level. We

assume that all the sentences in a document have the same sentiment. This is clearly an oversimplification, since some sentences (e.g., background) are sentiment neutral. Given that such sentences are more common in long reviews, we filter out reviews that exceed 10 sentences. We further filter the remaining sentences by eliminating those that exceed 15 words. The resulting dataset has 250K negative sentences, and 350K positive ones. The vocabulary size is 10K after replacing words occurring less than 5 times with the "<unk>" token. As a baseline model, we compare against the control-gen model of Hu et al. [51].

To quantitatively evaluate the transfered sentences, we adopt a model-based metric similar to the one used for image transfer [54]. Specifically, we measure how often a transferred sentence has the correct sentiment according to a pre-trained sentiment classifier. For this purpose, we use the TextCNN model as described in Kim [61]. On our simplified dataset for style transfer, it achieves nearly perfect accuracy of 97.4%.

While the quantitative evaluation provides some indication of transfer quality, it does not capture all the aspects of this generation task. Therefore, we also perform two human evaluations on 500 sentences randomly selected from the test set[1]. In the first evaluation, the judges were asked to rank generated sentences in terms of their fluency and sentiment. Fluency was rated from 1 (unreadable) to 4 (perfect), while sentiment categories were "positive", "negative", or "neither" (which could be contradictory, neutral or nonsensical). In the second evaluation, we evaluate the transfer process comparatively. The annotator was shown a source sentence and the corresponding outputs of the systems in a random order, and was asked "Which transferred sentence is semantically equivalent to the source sentence with an opposite sentiment?". They can be both satisfactory, A/B is better, or both unsatisfactory. We collect two labels for each question. The label agreement and conflict resolution strategy can be found in the supplementary material. Note that the two evaluations are not redundant. For instance, a system that always generates the same grammatically correct sentence with the right sentiment independently of the source sentence will score high in the first evaluation setup, but low in the second one.

[1] we eliminated 37 sentences from them that were judged as neutral by human judges.

Word Substitution Decipherment Our second set of experiments involves decipherment of word substitution ciphers, which has been previously explored in NLP literature [28, 90]. These ciphers replace every word in plaintext (natural language) with a cipher token according to a 1-to-1 substitution key. The decipherment task is to recover the plaintext from ciphertext. It is trivial if we have access to parallel data. However, we are interested to consider a non-parallel decipherment scenario. For training, we select 200K sentences as X_1, and apply a substitution cipher f on a different set of 200K sentences to get X_2. While these sentences are non-parallel, they are drawn from the same distribution from the review dataset. The development and test sets have 100K parallel sentences $D_1 = \{x^{(1)}, \ldots, x^{(n)}\}$ and $D_2 = \{f(x^{(1)}), \ldots, f(x^{(n)})\}$. We can quantitatively compare between D_1 and transferred (deciphered) D_2 using BLEU score [93].

Clearly, the difficulty of this decipherment task depends on the number of substituted words. Therefore, we report model performance with respect to the percentage of the substituted vocabulary. Note that the transfer models do not know that f is a word substitution function. They learn it entirely from the data distribution.

In addition to different transfer models, we introduce a simple decipherment baseline based on word frequency. Specifically, shared words between X_1 and X_2 do not require translation, the rest of the words are mapped based on their frequency, and ties are broken arbitrarily. Finally, to assess the difficulty of the task, we report the accuracy of a machine translation system trained on a parallel corpus [65].

Word Order Recovery Our final experiments focus on the word ordering task, also known as bag translation [13, 108]. By learning the style transfer functions between original English sentences X_1 and shuffled English sentences X_2, the model can be used to recover the original word order of a shuffled sentence (or conversely to randomly permute a sentence). The process to construct non-parallel training data and parallel testing data is the same as in the word substitution decipherment experiment. Again the transfer models do not know that f is a shuffle function and learn it completely from data.

Table 4.1: Sentiment accuracy of transferred sentences measured by a pretrained classifier.

Method	**ACC**
Hu et al. [51]	83.5
Variational autoencoder	23.2
Aligned autoencoder	48.3
Cross-aligned autoencoder	78.4

Table 4.2: Human evaluations on sentiment, fluency and overall transfer quality. Fluency rating is from 1 (unreadable) to 4 (perfect). Overall transfer quality is evaluated in a comparative manner, where the judge is shown a source sentence and two transferred sentences, and decides whether they are both good, both bad, or one is better.

Method	**Sentiment**	**Fluency**	**Overall transfer**
Hu et al. [51]	70.8	3.2	41.0
Cross-align	62.6	2.8	41.5

4.6 Results

Sentiment Modification Table 4.1 and Table 4.2 show the performance of various models for both human and automatic evaluation. The control-gen model of Hu et al. [51] performs better in terms of sentiment accuracy in both evaluations. This is not surprising as their generation is directly guided by a sentiment classifier. Their system also achieves higher fluency score. However, these gains do not translate into improvements in terms of the overall transfer, where our model faired better. As can be seen from the examples listed in Table 4.3, our model is more consistent with the grammatical structure and semantic meaning of the source sentence. In contrast, their model achieves sentiment change by generating an entirely new sentence which has little overlap with the source. The discrepancy between the two experiments demonstrates the crucial importance of developing appropriate evaluation measures to compare models for style transfer.

Table 4.3: Sentiment transfer samples. The first line is an input sentence, the second and third lines are the generated sentences after sentiment transfer by Hu et al. [51] and our cross-aligned autoencoder, respectively.

From negative to positive
consistently slow .
consistently good .
consistently fast .
my goodness it was so gross .
my husband 's steak was phenomenal .
my goodness was so awesome .
it was super dry and had a weird taste to the entire slice .
it was a great meal and the tacos were very kind of good .
it was super flavorful and had a nice texture of the whole side .
From positive to negative
i love the ladies here !
i avoid all the time !
i hate the doctor here !
my appetizer was also very good and unique .
my bf was n't too pleased with the beans .
my appetizer was also very cold and not fresh whatsoever .
came here with my wife and her grandmother !
came here with my wife and hated her !
came here with my wife and her son .

Table 4.4: BLEU scores of word substitution decipherment and word order recovery.

Method	**Substitution decipher**					**Order recover**
	20%	40%	60%	80%	100%	
No transfer (copy)	56.4	21.4	6.3	4.5	0	5.1
Unigram matching	74.3	48.1	17.8	10.7	1.2	-
Variational autoencoder	79.8	59.6	44.6	34.4	0.9	5.3
Aligned autoencoder	81.0	68.9	50.7	45.6	7.2	5.2
Cross-aligned autoencoder	**83.8**	**79.1**	**74.7**	**66.1**	**57.4**	**26.1**
Parallel translation	99.0	98.9	98.2	98.5	97.2	64.6

Word Substitution Decipherment Table 4.4 summarizes the performance of our model and the baselines on the decipherment task, at various levels of word substitution. Consistent with our intuition, the last row in this table shows that the task is trivial when the parallel data is provided. In non-parallel case, the difficulty of the task is driven by the substitution rate. Across all the testing conditions, our cross-aligned model consistently outperforms its counterparts. The difference becomes more pronounced as the task becomes harder. When the substitution rate is 20%, all methods do a reasonably good job in recovering substitutions. However, when 100% of the words are substituted (as expected in real language decipherment), the poor performance of variational autoencoder and aligned autoencoder rules out their application for this task.

Word Order Recovery The last column in Table 4.4 demonstrates the performance on the word order recovery task. Order recovery is much harder—even when trained with parallel data, the machine translation model achieves only 64.6 BLEU score. Note that some generated orderings may be completely valid (e.g., reordering conjunctions), but the models will be penalized for producing them. In this task, only the cross-aligned autoencoder achieves grammatical reorder to a certain extent, demonstrated by its BLEU score 26.1. Other models fail this task, doing no better than no transfer.

4.7 Conclusion

Transferring languages from one style to another has been previously trained using parallel data. In this work, we formulate the task as *a decipherment problem* with access only to non-parallel data. The two data collections are assumed to be generated by a latent variable generative model. Through this view, our method optimizes neural networks by forcing distributional alignment (invariance) over the latent space or sentence populations. We demonstrate the effectiveness of our method on tasks that permit quantitative evaluation, such as sentiment transfer, word substitution decipherment and word ordering. The decipherment view also provides an interesting open question—*when can the joint distribution* $p(x_1, x_2)$ *be recovered given only marginal distributions?* We believe addressing this general question would promote the style transfer research in both vision and NLP.

Chapter 5

Language Style Transfer with Confounders

5.1 Introduction

Despite advances in neural text generation [14], fine-grained control over generated outputs remains a significant challenge. Indeed, the ability to easily transfer output style by altering attributes such as sentiment, formality, genre, and personal styles would make text generation tools more appealing [52, 103, 110, 132].

Solving the problem of style transfer typically requires the method to be able to disentangle what should be transferred from orthogonal aspects of sentences that ought to be kept intact. This disentanglement problem can be largely avoided in simple supervised scenarios by giving the access to parallel sentences differing only in style (e.g., sentiment transfer with parallel negative and positive sentences, Figure 5-1.a). Recent approaches address a more difficult version of the task by dispensing with parallel sentences [110]. Nevertheless, they assume that corpora differing in style remain distributionally matched in other ways (e.g., sentiment transfer with negative and positive reviews from the same category of products, Figure 5-1.b).

However, data available for training style transfer models is rarely distributionally matched and often involves changes other than style as well (e.g., sentiment transfer where negative and positive reviews come from different product categories, Figure 5-

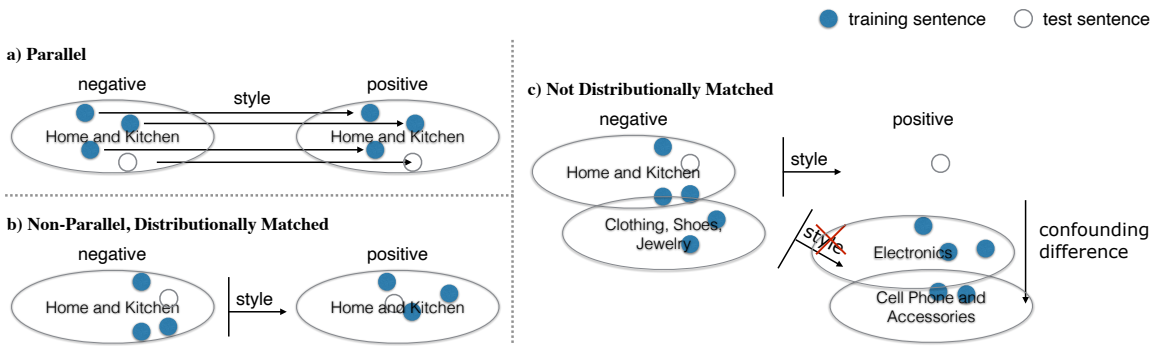

Figure 5-1: Different learning scenarios for style transfer. a) With parallel examples, learning of the transfer mapping is supervised. b) With non-parallel, distributionally matched datasets, learning of the transfer mapping is unsupervised. Nevertheless, style is given as the dataset difference, and generation takes place in-distribution. c) With non-parallel, not distributionally matched datasets, style needs to be inferred by excluding confounding differences. Not only is learning of the transfer mapping unsupervised, but generation is also out-of-distribution.

1.c). This mismatch means that the desired style difference is no longer illustrated directly as the difference between the two corpora, making the task substantially more challenging. More subtly, the model is asked to generate sentences it has not seen during training, since the generated sentences in a new style should not also reproduce those confounding differences present in the training data.

Solving style transfer with confounding cues requires us to infer what the desired style difference is. We show that this inference can be facilitated by dividing the data into two groups of different styles, while the sets within each group illustrate variations we do not wish to alter. In the example in Figure 5-1.c, reviews are divided into two groups according to their sentiment, and within a group, each dataset corresponds to a different product category which needs to be preserved. Besides sentiment transfer, our setup easily generalizes to other style transfer tasks, e.g., dialectal transfer. In this case, the groups will be divided according to dialect, and sets within each group will represent different speakers whose personal style should be preserved.

Our model builds on invariant risk minimization [3] to infer style as an invariant distinction across different datasets from the two groups. The resulting style classifier leaves complementary aspects of sentences to be controlled (preserved). We can illustrate aspects that are orthogonal to style with a new set of environments and learn

another invariant classifier. Together, the two classifiers are used to guide sentence generation in a style transfer model along the desired direction. Combining them with back-translation [109, 145] and language model regularization [47] techniques, we can generate new types of sentences that have not been seen during training.

We empirically evaluate our proposed model in two sentiment transfer settings that involve confounding factors. In the first setting, we augment original review data with special tokens, creating a spurious correlation with sentiment. In the second setting, we consider sentiment transfer in which negative and positive reviews come from non-overlapping product categories. In both cases, we assess the ability of the model to transfer sentiment while preserving other aspects: special tokens in the first case and product category in the second. Our experiments demonstrate that our model successfully achieves both tasks, bringing significant gains over baselines that do not consider confounding factors. For instance, on the the task of sentiment transfer from different product categories, the model yields a 28.4% increase in category preservation, and according to human evaluation, its success rate is 6.2% higher than the previous best system.

5.2 Related Work

The task of style transfer is related to paraphrasing, whose goal is to generate multiple linguistic realizations of the same underlying content [1]. However, style transfer adds an additional complexity – the requirement to control for a specific realization characteristic during rewriting. While paraphrasing models have been used in the past for the task of style transfer, these models are impacted by arbitrary variations present in paraphrasing datasets [42, 66, 97]. We instead take a data-driven approach for discerning style from content, where the style is not strictly limited to paraphrasing and can involve more general attribute transfer such as sentiment and political slant transfer [96].

Recent work in non-parallel style transfer proposes different techniques such as cross-alignment [110], delete and retrieval [71], or parallel latent sequences [47]. The

generation of these approaches all takes place "in-distribution", i.e., realizing sentences of the type already seen during training (Figure 5-1.b). The sentences we generate are by design not seen during training (Figure 5-1.c). Subramanian et al. [117] proposed multi-attribute style transfer that controls both the sentiment and the category of a sentence. However, similar to other prior work, they assume access to all sentiment-category combinations for training. In contrast, we assume that negative and positive sentences come from non-overlapping categories.

Since we have to infer style from datasets with confounding cues, we make use of invariant learning [3, 94] to estimate style as an invariant direction. Moreover, our classifiers need to perform well across different combinations of source/target datasets and generalize to generated text (out-of-distribution samples) to properly guide the transfer model. Therefore, our problem is also related to domain adaptation [9, 34, 35] and domain generalization [11, 86]. Our main contribution is the utilization of invariance for guiding the generation process.

5.3 Style Transfer with Confounders

We consider two groups of datasets: $G_A = \{A_1, \ldots, A_n\}$ where each A_i $(i = 1, \ldots, n)$ is a dataset consisting of sentences with style s_A (e.g., negative sentiment), and $G_B = \{B_{n+1}, \ldots, B_{n+m}\}$ where each B_j $(j = n+1, \ldots, n+m)$ similarly conforms to style s_B (e.g., positive sentiment). In addition to style, a dataset has its own characteristics different from each other (e.g., category). Our goal is to transfer a sentence x of style s_A into style s_B (and vice versa) without changing its content or other characteristics.

One attempt is to aggregate collections of sentences in each group $A = A_1 \cup \cdots \cup A_n, B = B_{n+1} \cup \cdots \cup B_{n+m}$ and perform style transfer between them. However, the specific characteristics of A_i and B_j will become confounding factors and will be changed along with style. Instead, we notice that style is an invariant distinction between group G_A and group G_B. In other words, the style difference is stable across different A_i and B_j. Therefore, we can learn to isolate it by taking out intra-group variations. Once we have access to style, we can learn to transfer sentences along this

direction while preserving other sentence characteristics.

We take a two-step procedure to accomplish this task. We first learn a pair of invariant classifiers to detect style and orthogonal characteristics. Then we use the classifiers to guide the learning of a style transfer model. Each part of the model is described in detail below.

5.3.1 Invariant Classifiers

Background

We make use of Invariant Risk Minimization [3, IRM] to learn our style and orthogonal classifiers. IRM requires us to specify a set of environments $\mathcal{E} = \{e_1, \ldots, e_E\}$, where each $e \in \mathcal{E}$ represents data $\{(x_i^e, y_i^e)\}_{i=1}^{n_e}$ collected under a certain environment. The different environments account for nuisance variation that the classifier should not pay attention to. The IRM objective is to learn a feature representation that enables a classifier to be simultaneously optimal for all environments. The rationale is that such representation likely involves primarily causal features that remain stable regardless of the nuisance variation. Therefore, the classifier can better generalize to new, unseen test environments compared to the standard Empirical Risk Minimization (ERM) classifier trained on the pooled data from all environments.

We adopt the IRMv1 formulation, which does not explicitly separate the representation from the classifier but treats the classifier output itself as the representation. In this vein, the objective becomes to minimize the classifier loss across all the data while penalizing per-environment gradients with respect to any multiplier of the classifier output:

$$\min_{\Phi:\mathcal{X}\to\mathcal{Y}} \sum_{e\in\mathcal{E}} R^e(\Phi) + \lambda \|\nabla_{w|w=1.0} R^e(w \cdot \Phi)\|^2 \tag{5.1}$$

where $R^e(f) := \mathbb{E}_{X^e,Y^e}[\ell(f(X^e), Y^e)]$ is the risk of f under environment e (ℓ can be any loss function), and λ is a hyperparameter weighting the gradient penalty term. Gradients would be zero if Φ is per-environment optimal. It remains to define suitable

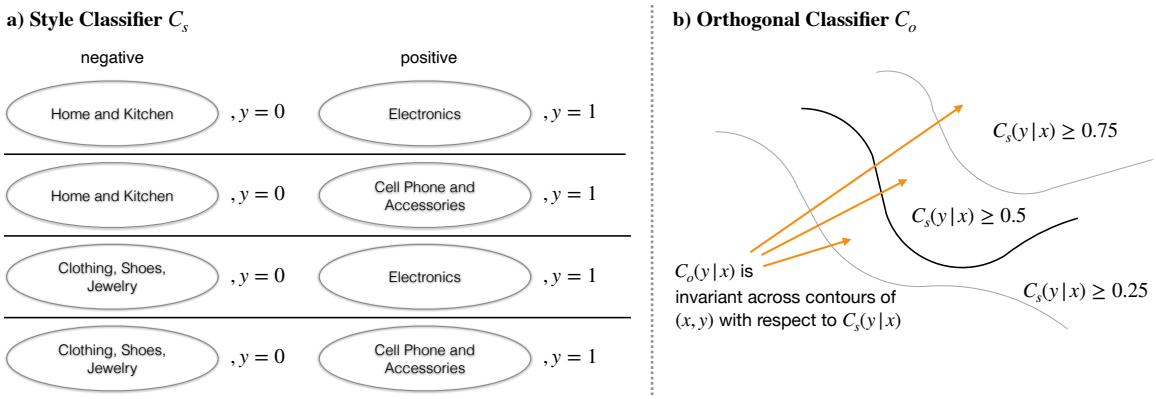

Figure 5-2: Illustration of the learning of the invariant classifiers. a) The style classifier C_s is trained to be invariant across different pairs of A_i and B_j datasets. b) The orthogonal classifier C_o is trained to highlight changes in sentences other than the style identified by C_s.

environments for the invariant classifiers to separate style and confounding factors in our task.

The Style Classifier

To learn a classifier to distinguish between styles s_A and s_B without relying the specific characteristics of each dataset, we pair each A_i and each B_j to form environments:

$$e_{i,j} = \{(x, y = 0) \mid x \in A_i\} \cup \{(x, y = 1) \mid x \in B_j\} \tag{5.2}$$

and learn an IRM classifier $C_s : \mathcal{X} \to \mathcal{Y}$ to predict the group label across environments $\{e_{i,j}\}_{1 \leq i \leq n, n+1 \leq j \leq n+m}$ (cf. Figure 5-2.a). Since all A_i datasets share style s_A, and all B_j datasets share style s_B, style is a feature representation that elicits an invariant classifier across different environments. Conversely, if the classifier uses any features specific to A_i/B_j, it will not be optimal in another environment consisting of a different pair $A_{i'}/B_{j'}$, thus violating the IRM constraint.

The Orthogonal Classifier

In addition to requiring the transferred output to have a different style from the input, we also require it to retain the other characteristics of the input. To detect

style-independent characteristics, we construct environments based on predictions of the style classifier C_s. These style-dependent environments serve to illustrate to the second invariant classifier C_o what it should not rely on.

Let $D = \{(x, y = 0) \mid x \in A\} \cup \{(x, y = 1) \mid x \in B\}$ be the entire dataset, aggregate of all corpora. We create two environments[1]:

$$e_1 = \{(x, y) \in D \mid C_s(y|x) > 0.5\} \tag{5.3}$$
$$e_2 = \{(x, y) \in D \mid C_s(y|x) \leq 0.5\}$$

and learn an IRM classifier $C_o : \mathcal{X} \to \mathcal{Y}$ across $\{e_1, e_2\}$. In this way, C_o cannot depend on the direction quantified by C_s (i.e., the inferred style), and must find other orthogonal features to distinguish A from B (cf. Figure 5-2.b).

Note that if we have successfully transferred a sentence in A_i to style s_B, then C_s should predict label 1 for the output sentence because its style has been changed, while C_o should continue to assign label 0 as the orthogonal characteristics ought to have remained intact.

5.3.2 The Style Transfer Model

Based on the pair of invariant classifiers, we can now build a style transfer model to transfer a sentence to a different style specified by C_s while preserving other characteristics controlled by C_o. Formally, we have dataset $D = \{(x, y)\}$, where y denotes the group label of sentence x. We learn a style transfer model $M : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}$ that takes a source sentence x and a target group y as input, and outputs a revised sentence that conforms to the style of group y. The model is learned first in a reconstruction phase and then in a transfer phase, discussed in turn.

When the input sentence x is from group y, M should behave as an autoencoder

[1] A set of more segmented environments can be created based on the confidence of C_s. However, we did not observe performance gains with more environments.

and reconstruct x. Therefore, we have the reconstruction loss:

$$\mathcal{L}_{\text{rec}}(\theta_M; x, y) = -\log p_M(x|x, y) \tag{5.4}$$

where θ_M denotes the parameters of M to be learned. We first use $\mathcal{L}_{\text{rec}}$ to train M on D for an epoch to provide the model with a good initialization to generate realistic sentences.

After initializing M as an autoencoder, we use the pair of invariant classifiers to guide it towards appropriate style transfer. Given an example (x, y) in D, we let $\tilde{x} \sim p_M(\cdot|x, 1-y)$ be the transferred output sampled from the model. If successfully transferred, $\tilde{x}$ should have style different from x according to the style classifier C_s, and all the other characteristics should be the same as x according to the orthogonal classifier C_o. Namely, C_s should treat $\tilde{x}$ as coming from a different group, and C_o should treat $\tilde{x}$ as coming from the same group. These two constraints lead to losses:

$$\mathcal{L}_{C_s}(\theta_M; y, \tilde{x}) = -\log p_{C_s}(1-y|\tilde{x}) \tag{5.5}$$

$$\mathcal{L}_{C_o}(\theta_M; y, \tilde{x}) = -\log p_{C_o}(y|\tilde{x}) \tag{5.6}$$

We further use a pre-trained language model L and introduce a KL divergence term $D_{\text{KL}}(p_M(\cdot|x, 1-y)\|p_L)$ to regularize the transfer distribution. Estimating the KL divergence with one sample of $\tilde{x} \sim p_M(\cdot|x, 1-y)$, we have:

$$\mathcal{L}_{\text{LM}}(\theta_M; x, y, \tilde{x}) = -\log p_L(\tilde{x}) + \log p_M(\tilde{x}|x, 1-y) \tag{5.7}$$

The first term ensures the fluency of the generated sentence $\tilde{x}$. The second term corresponds to the negative entropy of the transfer distribution $-H_{p_M(\cdot|x,1-y)}$. Maximizing this entropy term encourages exploration and helps to avoid bad local optima of constantly generating similar sentences [47].

Finally, we include the back-translation loss that the original sentence x should

be generated if we take $\tilde{x}$ as input and set the target label to be y [109, 145]:

$$\mathcal{L}_{\text{BT}}(\theta_M; x, y, \tilde{x}) = -\log p_M(x|\tilde{x}, y) \tag{5.8}$$

Taken together, our overall training objective is:

$$\mathbb{E}_{(x,y)\sim D, \tilde{x}\sim p_M(\cdot|x,1-y)}[\lambda_1\mathcal{L}_{C_s} + \lambda_2\mathcal{L}_{C_o} + \lambda_3\mathcal{L}_{\text{LM}} + \lambda_4\mathcal{L}_{\text{BT}}] \tag{5.9}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are hyperparameters to weight different loss terms. In practice, we use word-level loss for $\mathcal{L}_{\text{LM}}$ and $\mathcal{L}_{\text{BT}}$ (i.e., divided by the sentence length) to make them comparable in magnitude to $\mathcal{L}_{C_s}$ and $\mathcal{L}_{C_o}$. We use Gumbel-Softmax [55] to approximate the discrete sampling process of $\tilde{x}$ to compute gradients. At test time, we perform greedy decoding to generate the transferred output.

5.4 Experiments

To assess the ability of our model to perform style transfer in the presence of confounders, we consider two experimental settings. In the first experiment, we compose a synthetic task by modifying sentence punctuation to create a spurious correlation with sentiment [20]. The model needs to transfer the sentiment while preserving the punctuation. In the second experiment, we consider sentiment transfer across different product categories. The goal is to transfer sentiment without changing the product category.

Baselines We consider two variants of our full model: the first is guided by the style classifier C_s alone without the orthogonal classifier C_o; the second is guided by an ERM classifier C_{ERM} trained on D, performing direct transfer between A and B without taking into account confounders. We also compare with He et al. [47], whose training objective has a back-translation loss and a KL divergence term similar to ours. They do not use a classifier, but rely on two language models separately trained on A and B to promote the transferred sentence to have the target style. Finally,

we compare with a paraphrasing based method of Krishna et al. [66], which again transfers between the aggregated A and B without considering confounding factors.

Model Architecture We implement the classifiers C_s, C_o using TextCNN [60] with sliding windows over 3-5 words. On the synthetic task, however, we found that because the dataset is simple to classify, the training loss of the CNN classifier is close to 0. This removes any guidance from IRM constraints, degenerating the resulting classifier to a standard ERM version. Therefore, to ensure that we estimate invariant classifiers, we use less powerful Bag-of-Words classifiers in the synthetic task. The employed language model L is a 1-layer LSTM trained on $A \cup B$. The style transfer model M consists of a 1-layer LSTM encoder and a 1-layer LSTM decoder augmented with attention mechanism [7]. To achieve style control through y, we learn two style embeddings for $y = 0, y = 1$ respectively, and add the corresponding one to each word embedding before feeding to the decoder.

Training Regime and Hyperparameters During the training of the invariant classifiers, we linearly anneal the gradient penalty coefficient λ from 0 to $\lambda_{\max}$ over the first 100K steps, and continue training with $\lambda = \lambda_{\max}$ for another 200K steps. We try $\lambda_{\max}$ in $\{1000, 3000, 5000\}$ and choose the one with the best performance on the validation set. In the training of the style transfer model M, we linearly anneal the temperature of Gumbel-Softmax from 1 to 0.1 in the first 20K steps, and then keep it at 0.1 in the next 10K steps. We try the weights of the loss terms $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ in $\{1, 2, 4\}$, and select a model that strikes a good balance between output perplexity, BLEU with input, and accuracies according to the style and orthogonal classifiers. We note that different weight combinations will lead to different trade-offs [92], and we provide more detailed results in Appendix D.2.

5.4.1 Sentiment Transfer with Different Punctuation

We aim to emulate the presence of confounding factors by modifying sentences in a standard sentiment transfer dataset with special symbols. For a sentence x, we remove

its original punctuation, and then add either an exclamation mark "!" or a period "." at the end. Let p be the probability of adding "!", and $1 - p$ be the probability of adding ".". p varies across different corpora. This setting enables us to measure model's ability to transfer sentence sentiment without changing its punctuation.

Dataset We adapt the sentiment transfer dataset introduced by Shen et al. [110], which has 177K negative sentences and 267K positive sentences for training, 2K negative and positive sentences for validation, and 500 negative and positive sentences for testing. We obtain new training, validation and test sets using the following procedure: (1) negative sentences are modified with $p = 0$ to form A_1; (2) positive sentences are equally divided into two sets, and modified with $p = 1, p = 0.8$ to form B_2, B_3 respectively. By construction, the punctuation strongly correlates with the sentiment. Therefore, a direct transfer between A_1 and $B_2 \cup B_3$ is likely to change the punctuation as well. By observing that sentiment is invariant while p is variant in B_2 and B_3, we aim to transfer only the sentiment, not the punctuation.

Evaluations We assess transfer accuracy by comparing change in sentiment in the input and output sentences. To automate this evaluation, we utilize a separate sentiment classifier trained on negative and positive sentences both modified with $p = 0.5$. In this way, its prediction will not be affected by punctuation. Note that this classifier is only used for evaluation, not for training the style transfer model. We also evaluate the model's ability to keep punctuation intact during transfer by directly comparing input and output punctuation. To measure fluency, we report the perplexity of the output measured by an unbiased language model trained on sentences modified with $p = 0.5$. Finally, we compute the BLEU score of the output with respect to a human reference [71].

Results We first report performance of the invariant classifiers C_s and C_o in the setting reflecting their intended use in our style transfer model. We reverse the correlation between punctuation and sentiment labels to simulate style transferred sentences, and assess their ability to predict the desired aspects. Specifically, we test

Table 5.1: Classifier accuracy in the synthetic task when the spurious correlation is reversed.

Model	Sentiment	Punctuation
C_{ERM}	54.4	45.6
C_s	**75.3**	-
C_o	-	**89.9**

Table 5.2: Automatic evaluation results of the synthetic sentiment transfer task. Accuracies less than 30 are marked in red.

Model	Sentiment	Punctuation	PPL	BLEU_{ref}
Krishna et al. [66]	28.4	44.7	53.2	10.1
He et al. [47]	82.2	4.6	**27.0**	20.4
M w/ C_{ERM}	65.1	4.5	40.4	**28.8**
M w/ C_s	70.4	5.5	43.3	27.2
M w/ C_s, C_o (Ours)	**84.3**	**97.7**	48.1	24.3
Input Copy	2.4	100.0	34.0	32.8
Reference	76.8	100.0	42.3	100.0

the accuracy of classifiers on positive sentences modified with $p = 0$ and negative sentences modified with $p = 1, p = 0.8$. Table 5.1 shows results of C_s and C_o compared with the standard C_{ERM} classifier trained on D. As expected, C_{ERM} performs poorly, mixing sentiment and punctuation clues. In contrast, the invariant classifier successfully separating the two aspects, achieving accuracy of 75.3% on sentiment prediction and 89.9% on punctuation prediction.

Table 5.2 summarizes style transfer results. The paraphrasing-based method of Krishna et al. [66] is not suitable for sentiment transfer that requires semantic changes, as shown by its low sentiment accuracy of 28.4%. Moreover, it has the lowest BLEU score because it introduces unnecessary rewriting learned from the paraphrasing dataset. Despite reaching high sentiment accuracy, the direct transfer methods of He et al. [47] and M guided by C_{ERM} have low punctuation accuracy (4.6% and 4.5%, respectively). Our full model achieves both high sentiment accuracy (84.3%) and high punctuation accuracy (97.7%). The reliance on the orthogonal classifier C_o is proved critical – its omission results in dramatic drop of punctuation accuracy

Table 5.3: Example outputs of the synthetic sentiment transfer task.

Input	the sales people here are terrible .
Reference	the sales people are great .
Krishna et al.	the people here are absolutely terrible .
He et al.	the sales people here are great !
M w/ C_{ERM}	the sales people here are amazing !
M w/ C_s	the sales people here are fantastic !
Ours	the sales people here are amazing .
Input	great food but horrible staff and very very rude workers .
Reference	great food and excellent staff and very very nice workers .
Krishna et al.	great food , but very poor service .
He et al.	great food but excellent staff and very very friendly workers !
M w/ C_{ERM}	great food and great staff and very very nice workers !
M w/ C_s	great food but great staff and very very friendly workers !
Ours	great food and great staff and very very friendly workers .
Input	the food is delicious and plentiful !
Reference	the food was tough and dry !
Krishna et al.	the food is delicious and plentiful ! " .
He et al.	the food is mediocre and plentiful .
M w/ C_{ERM}	the food was mediocre and plentiful .
M w/ C_s	the food is mediocre and plentiful .
Ours	the food was mediocre , too !
Input	excellent combination of flavors , very unique !
Reference	the flavors are nothing to write home about !
Krishna et al.	very unique combination of flavors , very unique ! " .
He et al.	horrible customer service .
M w/ C_{ERM}	terrible combination of flavors , very disappointing .
M w/ C_s	terrible combination of flavors , not unique .
Ours	terrible combination of flavors , not outstanding !

(5.5%). Output examples in Table 5.3 provide multiple illustrations of these phenomena: Krishna et al. [66] often paraphrases the input without changing the sentiment, while other models consistently change punctuation; only our full model successfully transfers the sentiment without changing the punctuation.

5.4.2 Sentiment Transfer with Different Categories

Our second experiment focuses on sentiment transfer with product category as a confounding factor. Previous work took negative and positive reviews from the same category [71]. Our setting is more challenging where negative and positive reviews belong to distinct, non-overlapping categories. The model needs to infer that it is the sentiment to be transferred, not the category.

Dataset We use the 5-core Amazon review data [89], focusing on four large product categories: Clothing Shoes and Jewelry (CSJ), Home and Kitchen (HK), Electronics (E), and Cell Phone and Accessories (CPA). We further filter reviews based on their length, keeping reviews of length 5-20 words. Following standard practice, reviews with rating above three are considered positive, and those below three are considered negative. Reviews with rating three are discarded. Specifically, A_1 consists of negative reviews from the CSJ category, A_2 - negative reviews from HK, B_3 - positive reviews from E, and B_4 - positive from CPA. We create a dataset of 150K sentences for each category, in which 130K are used for training, 10K for validation and 10K for testing. Note that we do not use any review data of different sentiments from the same category.

Evaluations To assess sentiment transfer accuracy automatically, we train a classifier on complete data which includes negative and positive reviews from all four categories. Its test accuracy is 96.0%. Moreover, we train a 4-way category classifier using complete data to assess whether the output preserves input product category. It has test accuracy[2] 71.0%. As in the previous task, we report the perplexity of the

[2]This accuracy can be attributed to the fact that some short sentences do not have clear indicators of their product category.

Table 5.4: Automatic evaluation results of sentiment transfer from different categories.

Model	Sentiment	Category	PPL	BLEU_{src}
Krishna et al. [66]	22.6	**57.4**	**35.8**	19.2
He et al. [47]	77.7	22.5	44.6	47.6
M w/ C_{ERM}	**89.6**	14.6	42.4	47.0
M w/ C_s	78.0	36.4	45.1	**59.2**
M w/ C_s, C_o (Ours)	79.9	50.9	49.2	57.4
Input Copy	3.1	75.3	34.5	100.0

Table 5.5: Human evaluation results of sentiment transfer from different categories.

Model	Sentiment	Content	Fluency	Success
Krishna et al. [66]	1.9	**4.2**	**4.4**	11.8%
He et al. [47]	3.3	3.6	4.0	24.3%
Ours	**3.4**	4.1	4.1	**30.5%**

output measured by an unbiased language model trained on the complete data. Note that these omniscient models are used only for evaluation and not for training the style transfer model. Since we do not have human references for this task, we compute the BLEU score with respect to source sentences as a rough indicator of content preservation [143]. In addition, we conduct human evaluation by asking human judges to rate each output on a Likert scale from 1 to 5 on three criteria: sentiment transfer, content preservation, and fluency. We consider a generated output successful if it is rated 4 or 5 on all three criteria. We evaluated 200 randomly sampled examples (100 negative and 100 positive) and collected two annotations for each sentence.

Results Table 5.4 shows automatic evaluation results for different models. Our model yields significant performance gains over the baselines, achieving 57.3% absolute increase in sentiment transfer compared to Krishna et al. [66], and 28.4% increase in category preservation compared to He et al. [47]. The human evaluation results in Table 5.5 further verify the superiority of our model. Krishna et al. [66] scores the highest in fluency as their model is fine-tuned from GPT2-Large. Our model achieves the highest sentiment transfer score and the highest success rate.

Table 5.6: Example outputs of sentiment transfer from different categories.

Clothing, Shoes and Jewelry (negative → positive)	
Input	this shirt was too tight . the sizing seems off .
Krishna et al.	the shirt is too tight .
He et al.	this case was great . the protection seems great .
Ours	this shirt works just perfect . the sizing seems well .
Home and Kitchen (negative → positive)	
Input	the containers do not lock well and are made of low quality materials .
Krishna et al.	the containers do not fit securely and are made from poor quality material .
He et al.	the phones work well and has made of sound quality of low quality materials .
Ours	the containers does the job well and are made of high quality materials .
Electronics (positive → negative)	
Input	exactly as advertised . converted a molex plug into a sata
Krishna et al.	the molex plug was convert to sata as advertised .
He et al.	way too big . leaves a inaccurate cut into a bath
Ours	not as advertised . converted a molex plug into a sata
Cell Phones and Accessories (positive → negative)	
Input	very sturdy and helpful to use while driving .
Krishna et al.	drives very well and is very useful .
He et al.	very stiff and weak to use while washing .
Ours	very thin and uncomfortable to use while driving .

Example outputs in Table 5.6 show that our model successfully isolates sentiment from product category, preserving the latter during transfer. In contrast, He et al. [47] mixes up categories by changing product specific nouns, such as rewriting "shirt" to "case", "containers" to "phones", and "driving" to "washing". Table D.3.4 in the appendix provide more examples, some of which illustrate several failure modes of the model. For instance, when transferring sentiment of the sentence "very poor quality . crooked on one end .", the model only modifies the sentiment of the first clause, leaving the second clause intact. Another failure case is the use of inappropriate adjectives, such as rewriting "the drive works as designed" to "the drive is too large".

5.5 Conclusion

In this paper, we consider a new, challenging version of the style transfer task, where the model has to exclude confounders and infer the desired transfer direction from data. We propose to learn a pair of invariant classifiers to detect style and orthogonal characteristics, and then use them to guide a style transfer model. We empirically demonstrate significant performance gains over direct style transfer in two experimental settings. While this technology shows significant promise, it still requires further development to be used in practice. However, as style transfer algorithms continue to advance, their ability to support rapid and large-scale content modification will increase, which may promote the dissemination of fake news and other forms of misinformation.

Chapter 6

Conclusion

In this thesis, we have presented a serious of models and techniques to control language generation. First, we proposed the blank language model that supports generation location control. By dynamically filling and expanding blanks, the model can take account of the entire context during generation without the need for complex inference algorithms. Then, we looked at text autoencoders with latent variable control. We conducted in-depth theoretical and experimental analyses of the latent space geometry of text autoencoders, and showed that denoising promotes neighborhood preservation, enabling meaningful text manipulation through latent space operations. Finally, we studied an important application of controlling language style. A key challenge is the lack of supervised data. We first leveraged distributional alignment to perform style transfer from non-parallel data, and then further accounted for confounders in the data and exploited invariance to isolate them. Our approaches have demonstrated success on various controlled generation tasks, including text filling, ancient text restoration, sentence interpolation, tense transfer, and sentiment transfer.

As text generation models continue to advance and become more widely used, we must be very mindful of the impact they may have on society. We now provide some future research directions on fairness and privacy that follow from this thesis:

Future Research

Fairness Biases in training data can cause language models to generate biased content. For example, given the prompt "The doctor was a", GPT-3 will continue it with "man" with a much higher probability than "woman", and the opposite when prompted with "The nurse was a" [14]. Such biases in models will entrench existing stereotypes that could harm people in the relevant groups.

One way to eliminate this bias is to require the same number of female and male sentences in sentences containing "doctor". Conditioning on the presence of "doctor" can be readily expressed as "__ doctor __" in a blank language model. We can add fairness constraints to the training of blank language models to ensure fair generation.

Another interesting research direction is to use invariant learning discussed in Chapter 5 to study which words have time-varying correlations with gender and which words have invariant correlations. For example, "engineer" may have a strong correlation with gender, but this correlation is variant, as we now have more and more female engineers. In contrast, gender-specific words like "boys" and "girls" as well as physiological features like "beard" would have invariant correlations. This study can also help us understand how people's perceptions of gender change over time, and perhaps help us spot persistent gender biases that do not improve.

Privacy While better language models can be trained by aggregating data together, in many cases users may want to keep their personal data such as chat logs private. One possible way to achieve privacy is to share encodings rather than raw text data. So each user trains a private autoencoder separately and sends the encodings to a central server, the server trains a conditional generative model on all the encodings, and then each user decodes the model using their own decoder. We hope that the encodings still preserve most of the structure of language so that the server can benefit from more data to train a better generative model. Meanwhile, the encodings should be difficult to revert to plain text and should not leak too much information. We hypothesize that our denoising text autoencoder has the potential for this purpose.

Appendix A

Blank Language Models

A.1 Implementation Details for Text Infilling Baselines

A.1.1 Insertion Transformer

We implement the Insertion Transformer in our own framework, using the same Transformer encoder module as for BLM and replacing the prediction layers by Insertion Transformer's mechanism. The canvas is also generated according to the training procedure of Insertion Transformer.

A.1.2 Masked Language Model

We use the `RobertaForMaskedLM` architecture in the Transformers library for MLM [76, 127].

At test time, the model is given an easier version of the text infilling task where blanks are expanded into sequences of ⟨mask⟩ tokens of the target length (or equivalently, the model uses an oracle to predict the length of the infilling).

We experiment with three decoding strategies: (1) one-shot: the model predicts all masks simultaneously (2) left-to-right: the model fills in the masks from left to right (3) confident-first: the model fills one mask at a time that has the highest score.

We report results for the confident-first strategy which has the best performance.

A.1.3 BERT+LM

We use the `bert-base-uncased` model as served by the Transformers library [24, 127]. The left-to-right language model is a Transformer decoder to predict tokens in a blank. Its input word embedding is concatenated with BERT's output in the blank position at each time step.

A.1.4 Seq2seq-full and Seq2seq-fill

For both seq2seq baselines, we use Fairseq's `transformer_iwslt_de_en` architecture [91]. To generate training data, we apply the blanking procedure to the input dataset and generate k copies of each sentence with different masks. We experiment with $k \in \{1, 10, 100\}$ and report the best performance, obtained by $k = 10$.

A.2 Monte-Carlo Estimate of Perplexity

For a sentence x of length n, we estimate $p(x;\theta)$ in Eq. (2.5) with m samples:

$$X_m = \frac{n!}{m} \sum_{i=1}^{m} p(x, \sigma_i; \theta)$$

where σ_i's are randomly sampled orders.

Note that X_m is an unbiased estimate of $p(x;\theta)$:

$$\mathbb{E}[X_m] = p(x;\theta)$$

The estimated PPL is accordinly:

$$Y_m = X_m^{-1/n}$$

Since $z^{-1/n}$ is a convex function of z,

$$\mathbb{E}[Y_m] = \mathbb{E}[X_m^{-1/n}] \geq \mathbb{E}[X_m]^{-1/n} = p(x;\theta)^{-1/n}$$

i.e., the expectation of the estimated PPL $\geq$ the actual PPL. As m increases, the variance of X_m decreases, and the inequality becomes tighter.

Hence, we will observe that as m increases, the estimated PPL becomes smaller and converges to the real PPL.

A.3 Generation Trajectory

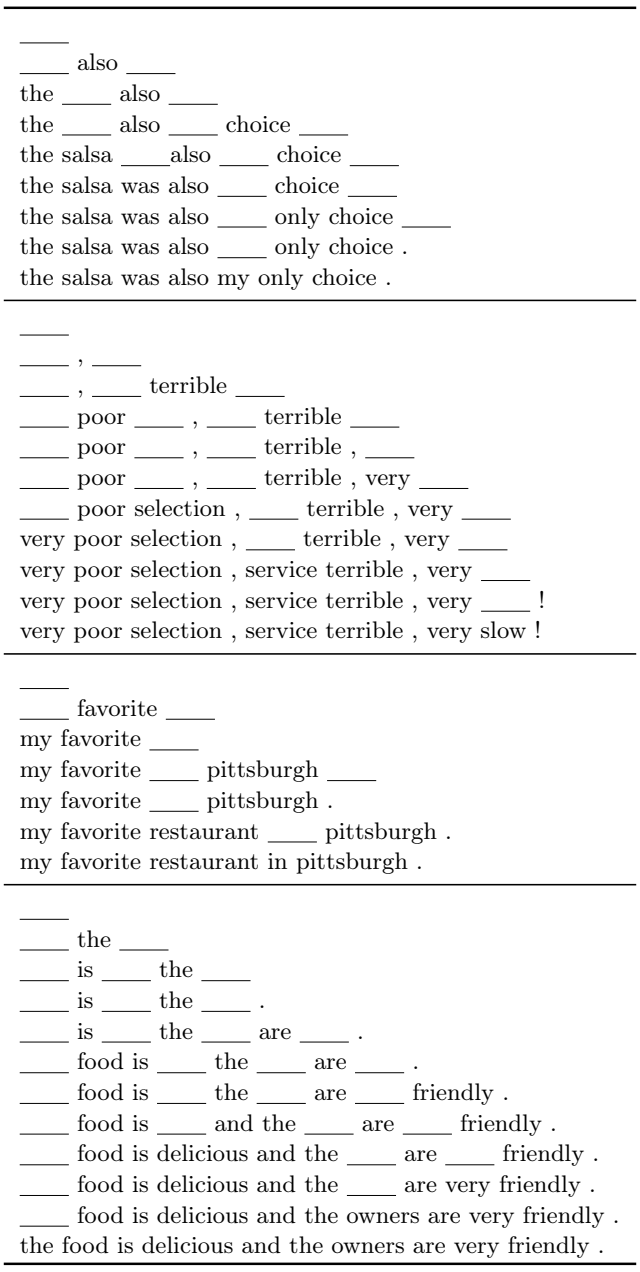

Figure A-1: Examples of BLM generation trajectory on the Yelp review dataset.

Appendix B

Denoising Text Autoencoders

B.1 Wasserstein Distance

The AAE objective can be connected to a relaxed form of the Wasserstein distance between model and data distributions [121]. Specifically, for cost function $c(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and deterministic decoder mapping $G : \mathcal{Z} \to \mathcal{X}$, it holds that:

$$\begin{aligned} & \inf_{\Gamma \in \mathcal{P}(x \sim p_{\text{data}}, y \sim p_G)} \mathbb{E}_{(x,y)\sim\Gamma}[c(x, y)] \\ = & \inf_{q(z|x):q(z)=p(z)} \mathbb{E}_{p_{\text{data}}(x)} \mathbb{E}_{q(z|x)}[c(x, G(z))] \end{aligned} \tag{B.1}$$

where the minimization over couplings Γ with marginals p_{data} and p_G can be replaced with minimization over conditional distributions $q(z|x)$ whose marginal $q(z) = \mathbb{E}_{p_{\text{data}}(x)}[q(z|x)]$ matches the latent prior distribution $p(z)$. Relaxing this marginal constraint via a divergence penalty $D(q(z)\|p(z))$ estimated by adversarial training, one recovers the AAE objective (Eq. 3.1). In particular, AAE on discrete x with the cross-entropy loss is minimizing an upper bound of the total variation distance between p_{data} and p_G, with c chosen as the indicator cost function [143].

Our model is optimizing over conditional distributions $q(z|x)$ of the form (3.6), a subset of all possible conditional distributions. Thus, after introducing input perturbations, our method is still minimizing an upper bound of the Wasserstein distance between p_{data} and p_G described in (B.1).

B.2 Toy Experiments with Latent Dimension 5

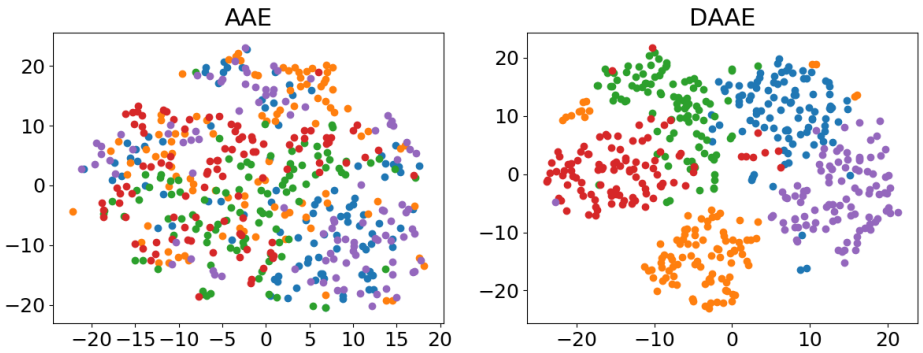

Figure B.2.1: t-SNE visualization of 5-D latent representations learned by AAE and DAAE when mapping clustered sequences in $\mathcal{X} = \{0,1\}^{50}$ to $\mathcal{Z} = \mathbb{R}^5$. The training data stem from 5 underlying clusters, with 100 sequences sampled from each (colored accordingly by cluster identity).

B.3 Proofs

Theorem 1. *For any one-to-one mapping E from $\{x_1, \ldots, x_n\}$ to $\{z_1, \ldots, z_n\}$, the optimal value of objective $\max_{G \in \mathcal{G}_L} \frac{1}{n} \sum_{i=1}^{n} \log p_G(x_i | E(x_i))$ is the same.*

Proof. Consider two encoder matchings x_i to $z_{\alpha(i)}$ and x_i to $z_{\beta(i)}$, where both α and β are permutations of the indices $\{1, \ldots, n\}$. Suppose G_α is the optimal decoder model for the first matching (with permutations α). This implies

$$p_{G_\alpha} = \arg\max_{G \in \mathcal{G}_L} \sum_{i=1}^{n} \log p_G(x_i | z_{\alpha(i)})$$

Now let $p_{G_\beta}(x_i | z_j) = p_{G_\alpha}(x_{\beta\alpha^{-1}(i)} | z_j), \forall i, j$. Then G_β can achieve exactly the same log-likelihood objective value for matching β as G_α for matching α, while still respecting the Lipschitz constraint. □

Theorem 2. *Let d be a distance metric over $\mathcal{X}$. Suppose x_1, x_2, x_3, x_4 satisfy that with some $\epsilon > 0$: $d(x_1, x_2) < \epsilon$, $d(x_3, x_4) < \epsilon$, and $d(x_i, x_j) > \epsilon$ for all other (x_i, x_j) pairs. In addition, z_1, z_2, z_3, z_4 satisfy that with some $0 < \delta < \zeta$: $\|z_1 - z_2\| < \delta$, $\|z_3 - z_4\| < \delta$, and $\|z_i - z_j\| > \zeta$ for all other (z_i, z_j) pairs. Suppose our perturbation process C reflects local $\mathcal{X}$ geometry with: $p_C(x_i|x_j) = 1/2$ if $d(x_i, x_j) < \epsilon$ and $= 0$ otherwise. For $\delta < 1/L \cdot (2\log(\sigma(L\zeta)) + \log 2)$ and $\zeta > 1/L \cdot \log\left(1/(\sqrt{2}-1)\right)$, the denoising objective $\max_{G \in \mathcal{G}_L} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} p_C(x_j|x_i) \log p_G(x_i|E(x_j))$ (where $n = 4$) achieves the largest value when encoder E maps close pairs of x to close pairs of z.*

Proof. Let $[n]$ denote $\{1, \ldots, n\}$, and assume without loss of generality that the encoder E maps each x_i to z_i. We also define $A = \{1, 2\}, B = \{3, 4\}$ as the two x-pairs that lie close together. For our choice of $C(x)$, the training objective to be maximized is:

$$\begin{aligned} &\sum_{i,j \in A} \log p_G(x_i|E(x_j)) + \sum_{k,\ell \in B} \log p_G(x_k|E(x_\ell)) \\ &= \sum_{i,j \in A} \log p_G(x_i|z_j) + \sum_{k,\ell \in B} \log p_G(x_k|z_\ell) \end{aligned} \tag{B.2}$$

The remainder of our proof is split into two cases:

Case 1. $||z_j - z_\ell|| > \zeta$ for $j \in A, \ell \in B$

Case 2. $||z_j - z_\ell|| < \delta$ for $j \in A, \ell \in B$

Under Case 1, x points that lie far apart also have z encodings that remain far apart. Under Case 2, x points that lie far apart have z encodings that lie close together. We complete the proof by showing that the achievable objective value in Case 2 is strictly worse than in Case 1, and thus an optimal encoder/decoder pair would avoid the x, z matching that leads to Case 2.

In Case 1 where $||z_j - z_\ell|| > \zeta$ for all $j \in A, \ell \in B$, we can lower bound the training objective (B.2) by choosing:

$$p_G(x_i|z_j) = \begin{cases} (1-\gamma)/2 & \text{if } i, j \in A \text{ or } i, j \in B \\ \gamma/2 & \text{otherwise} \end{cases} \tag{B.3}$$

with $\gamma = \sigma(-L\zeta) \in (0, \frac{1}{2})$, where $\sigma(\cdot)$ denotes the sigmoid function. Note that this ensures $\sum_{i \in [4]} p_G(x_i|z_j) = 1$ for each $j \in [4]$, and does not violate the Lipschitz condition from Assumption 2 since:

$$|\log p_G(x_i|z_j) - \log p_G(x_i|z_\ell)|$$

$$\begin{cases} = 0 & \text{if } j, \ell \in A \text{ or } j, \ell \in B \\ \leq \log\left((1-\gamma)/\gamma\right) & \text{otherwise} \end{cases}$$

and thus remains $\leq L||z_j - z_\ell||$ when $\gamma = \sigma(-L\zeta) \geq \sigma(-L||z_j - z_\ell||) = 1/[1 + \exp(L||z_j - z_\ell||)]$.

Plugging the $p_G(x|z)$ assignment from (B.3) into (B.2), we see that an optimal decoder can obtain training objective value $\geq 8 \log\left[\sigma(L\zeta)/2\right]$ in Case 1 where $||z_j - z_\ell|| > \zeta,\ \forall j \in A, \ell \in B$.

Next, we consider the alternative case where $||z_j - z_\ell|| < \delta$ for $j \in A, \ell \in B$.

For $i, j \in A$ and for all $\ell \in B$, we have:

$$\begin{aligned} \log p_G(x_i|z_j) &\leq \log p_G(x_i|z_\ell) + L||z_j - z_\ell|| && \text{(by Assumption 2)} \\ &\leq \log p_G(x_i|z_\ell) + L\delta \\ &\leq L\delta + \log\left[1 - \sum_{k \in B} p_G(x_k|z_\ell)\right] && \text{(since } \textstyle\sum_k p_G(x_k|z_\ell) \leq 1\text{)} \end{aligned}$$

Continuing from (B.2), the overall training objective in this case is thus:

$$
\begin{aligned}
& \sum_{i,j\in A} \log p_G(x_i|z_j) + \sum_{k,\ell\in B} \log p_G(x_k|z_\ell) \\
\leq\ & 4L\delta + \sum_{i,j\in A} \min_{\ell\in B} \log \left[1 - \sum_{k\in B} p_G(x_k|z_\ell)\right] \\
& + \sum_{k,\ell\in B} \log p_G(x_k|z_\ell) \\
\leq\ & 4L\delta + \sum_{\ell\in B} \left[2\log\left(1 - \sum_{k\in B} p_G(x_k|z_\ell)\right)\right. \\
& \left. + \sum_{k\in B} \log p_G(x_k|z_\ell)\right] \\
\leq\ & 4L\delta - 12\log 2
\end{aligned}
$$

using the fact that the optimal decoder for the bound in this case is: $p_G(x_k|z_\ell) = 1/4$ for all $k, \ell \in B$.

Finally, plugging our range for δ stated in the Theorem 2, it shows that the best achievable objective value in Case 2 is strictly worse than the objective value achievable in Case 1. Thus, the optimal encoder/decoder pair under the AAE with perturbed x will always prefer the matching between $\{x_1, \ldots, x_4\}$ and $\{z_1, \ldots, z_4\}$ that ensures nearby x_i are encoded to nearby z_i (corresponding to Case 1). □

Theorem 3. *Suppose $x_1, \ldots, x_n$ are divided into n/K clusters of equal size K, with S_i denoting the cluster index of x_i. Let the perturbation process C be uniform within clusters, i.e. $p_C(x_i|x_j) = 1/K$ if $S_i = S_j$ and $= 0$ otherwise. With a one-to-one encoder mapping E from $\{x_1, \ldots, x_n\}$ to $\{z_1, \ldots, z_n\}$, the denoising objective $\max_{G \in \mathcal{G}_L} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} p_C(x_j|x_i) \log p_G(x_i|E(x_j))$ has an upper bound: $(1/n^2) \cdot \sum_{i,j:S_i \neq S_j} \log \sigma(L\|E(x_i) - E(x_j)\|) - \log K$.*

Proof. Without loss of generality, let $E(x_i) = z_i$ for notational convenience. We consider what is the optimal decoder probability assignment $p_G(x_i|z_j)$ under the Lipschitz constraint 2.

The objective of the AAE with perturbed x is to maximize:

$$\begin{aligned} & \frac{1}{n} \sum_i \sum_j p_C(x_j|x_i) \log p_G(x_i|E(x_j)) \\ = \; & \frac{1}{nK} \sum_j \sum_{i:S_i=S_j} \log p_G(x_i|z_j) \end{aligned}$$

We first show that the optimal $p_G(\cdot|\cdot)$ will satisfy that the same probability is assigned within a cluster, i.e. $p(x_i|z_j) = p(x_k|z_j)$ for all i, k s.t. $S_i = S_k$. If not, let $P_{sj} = \sum_{i:S_i=s} p_G(x_i|z_j)$, and we reassign $p_{G'}(x_i|z_j) = P_{S_i j}/K$. Then G' still conforms to the Lipschitz constraint if G meets it, and G' will have a larger target value than G.

Now let us define $P_j = \sum_{i:S_i=S_j} p_G(x_i|z_j) = K \cdot p_G(x_j|z_j)$ $(0 \leq P_j \leq 1)$. The objective becomes:

$$\begin{aligned} & \max_{p_G} \frac{1}{nK} \sum_j \sum_{i:S_i=S_j} \log p_G(x_i|z_j) \\ = \; & \max_{p_G} \frac{1}{n} \sum_j \log p_G(x_j|z_j) \\ = \; & \max_{p_G} \frac{1}{n} \sum_j \log P_j - \log K \\ = \; & \max_{p_G} \frac{1}{2n^2} \sum_i \sum_j (\log P_i + \log P_j) - \log K \\ \leq \; & \frac{1}{2n^2} \sum_i \sum_j \max_{p_G} (\log P_i + \log P_j) - \log K \end{aligned}$$

Consider each term $\max_{p_G}(\log P_i + \log P_j)$: when $S_i = S_j$, this term can achieve the maximum value 0 by assigning $P_i = P_j = 1$; when $S_i \neq S_j$, the Lipschitz constraint ensures that:

$$\log(1 - P_i) \geq \log P_j - L\|z_i - z_j\|$$
$$\log(1 - P_j) \geq \log P_i - L\|z_i - z_j\|$$

Therefore:

$$\log P_i + \log P_j \leq 2 \log \sigma(L\|z_i - z_j\|)$$

Overall, we thus have:

$$\begin{aligned} & \max_{p_G} \frac{1}{nK} \sum_j \sum_{i:S_i=S_j} \log p_G(x_i|z_j) \\ \leq \; & \frac{1}{n^2} \sum_{i,j:S_i \neq S_j} \log \sigma(L\|z_i - z_j\|) - \log K \end{aligned}$$

□

B.4 Experimental Details

We use the same architecture to implement all models with different objectives. The encoder E, generator G, and the language model used to compute Forward/Reverse PPL are one-layer LSTMs with hidden dimension 1024 and word embedding dimension 512. The last hidden state of the encoder is projected into 128/256 dimensions to produce the latent code z for Yelp/Yahoo datasets respectively, which is then projected and added with input word embeddings fed to the generator. The discriminator D is an MLP with one hidden layer of size 512. λ of AAE based models is set to 10 to ensure the latent codes are indistinguishable from the prior. All models are trained via the Adam optimizer [63] with learning rate 0.0005, $\beta_1 = 0.5$, $\beta_2 = 0.999$. At test time, encoder-side perturbations are disabled, and we use greedy decoding to generate x from z.

B.5 Human Evaluation

For the tense transfer experiment, the human annotator is presented with a source sentence and two outputs (one from each approach, presented in random order) and asked to judge which one successfully changes the tense while being faithful to the source, or whether both are good/bad, or if the input is not suitable to have its tense inverted. We collect labels from two human annotators and if they disagree, we further solicit a label from the third annotator.

B.6 Neighborhood Preservation

Here we include non-generative models AE, DAE and repeat the neighborhood preservation analysis in Section 3.5.1. We find that an untrained RNN encoder from random initialization has a good recall rate, and we suspect that SGD training of vanilla AE towards only the reconstruction loss will not overturn this initial bias. Note that denoising still improves neighborhood preservation in this case. Also note that DAAE has the highest recall rate among all generative models that have a latent prior imposed.

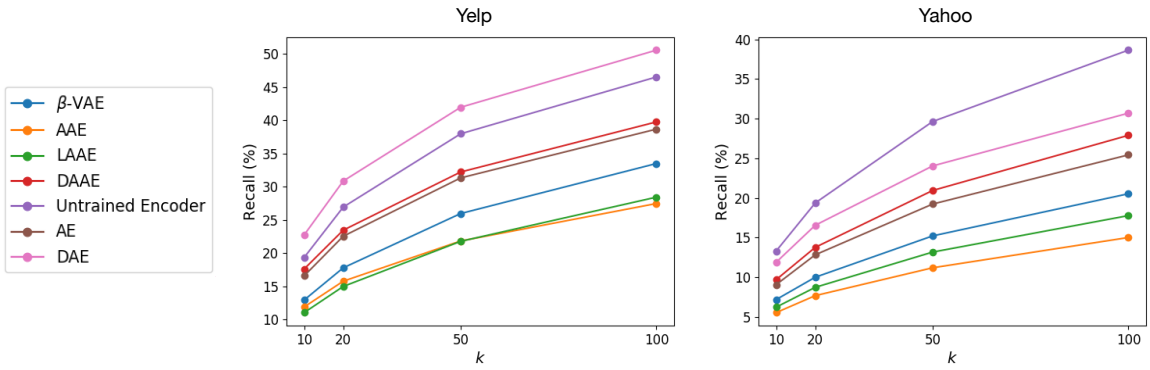

Figure B.6.2: Recall rate of 10 nearest neighbors in the sentence space retrieved by k nearest neighbors in the latent space of different autoencoders on the Yelp and Yahoo datasets.

Table B.6.1: Examples of nearest neighbors in the latent Euclidean space of AAE and DAAE on Yahoo dataset.

Source	**how many gospels are there that were n't included in the bible ?**
AAE	there are no other gospels that were n't included in the bible .
	how many permutations are there for the letters in the word _UNK ' ?
	anyone else picked up any of the _UNK in the film ?
	what 's the significance of the number 40 in the bible ?
	how many pieces of ribbon were used in the _UNK act ?
DAAE	there are no other gospels that were n't included in the bible .
	how many litres of water is there in the sea ?
	how many _UNK gods are there in the classroom ?
	how many pieces of ribbon were used in the _UNK act ?
	how many times have you been grounded in the last year ?
Source	**how do i change colors in new yahoo mail beta ?**
AAE	how should you present yourself at a _UNK speaking exam ?
	how can i learn to be a hip hop producer ?
	how can i create a _UNK web on the internet ?
	how can i change my _UNK for female not male ?
	what should you look for in buying your first cello ?
DAAE	how do i change that back to english ?
	is it possible to _UNK a yahoo account ?
	how do i change my yahoo toolbar options ?
	what should you look for in buying your first cello ?
	who do you think should go number one in the baseball fantasy draft , pujols or _UNK ?

B.7 Generation-Reconstruction Results on the Yahoo Dataset

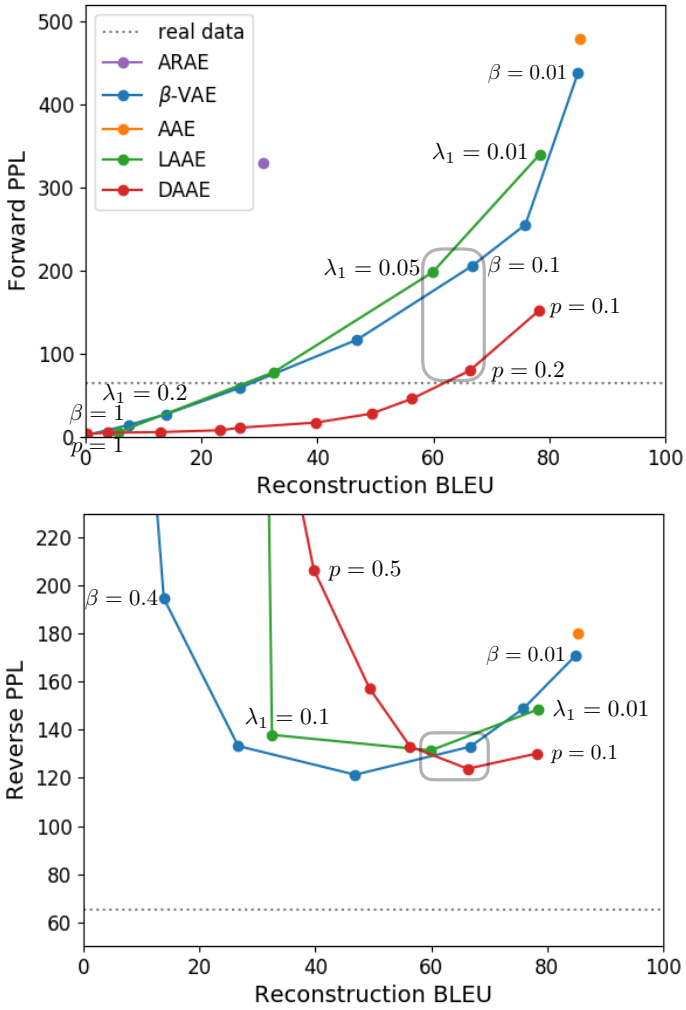

Figure B.7.3: Generation-reconstruction trade-off of various text autoencoders on the Yahoo dataset. The "real data" line marks the PPL of a language model trained and evaluated on real data. We strive to approach the lower right corner with both high BLEU and low PPL. The grey box identifies hyperparameters we finalize for respective models. Points of severe collapse (Reverse PPL > 300) are removed from the right panel.

B.8 Additional Examples

Table B.8.2: Additional examples of vector arithmetic for tense inversion.

Input	**the staff is rude and the dr. does not spend time with you .**
ARAE	the staff is rude and the dr. does not worth two with you .
β-VAE	the staff was rude and the dr. did not spend time with your attitude .
AAE	the staff was rude and the dr. does not spend time with you .
LAAE	the staff was rude and the dr. is even for another of her entertained .
DAAE	the staff was rude and the dr. did not make time with you .
Input	**slow service , the food tasted like last night 's leftovers .**
ARAE	slow service , the food tasted like last night 's leftovers .
β-VAE	slow service , the food tastes like last place serves .
AAE	slow service , the food tasted like last night 's leftovers .
LAAE	slow service , the food , on this burger spot !
DAAE	slow service , the food tastes like last night ... .
Input	**they are the worst credit union in arizona .**
ARAE	they are the worst bank credit in arizona .
β-VAE	they were the worst credit union in my book .
AAE	they are the worst credit union in arizona .
LAAE	they were the worst credit union in my heart .
DAAE	they were the worst credit union in arizona ever .
Input	**i reported this twice and nothing was done .**
ARAE	i swear this twice and nothing was done .
β-VAE	i 've gone here and nothing too .
AAE	i reported this twice and nothing was done .
LAAE	i dislike this twice so pleasant guy .
DAAE	i hate this pizza and nothing done .

Table B.8.3: Additional examples of vector arithmetic for sentiment transfer.

Input		**this woman was extremely rude to me .**
AAE	$+v$	this woman was extremely rude to me .
	$+1.5v$	this woman was extremely rude to baby .
	$+2v$	this woman was extremely rude to muffins .
DAAE	$+v$	this woman was extremely nice .
	$+1.5v$	this staff was amazing .
	$+2v$	this staff is amazing .
Input		**my boyfriend said his pizza was basic and bland also .**
AAE	$+v$	my boyfriend said his pizza was basic and tasty also .
	$+1.5v$	my shared said friday pizza was basic and tasty also .
	$+2v$	my shared got pizza pasta was basic and tasty also .
DAAE	$+v$	my boyfriend said his pizza is also excellent .
	$+1.5v$	my boyfriend and pizza is excellent also .
	$+2v$	my smoked pizza is excellent and also exceptional .
Input		**the stew is quite inexpensive and very tasty .**
AAE	$-v$	the stew is quite inexpensive and very tasty .
	$-1.5v$	the stew is quite inexpensive and very very tasteless .
	$-2v$	the – was being slow - very very tasteless .
DAAE	$-v$	the stew is quite an inexpensive and very large .
	$-1.5v$	the stew is quite a bit overpriced and very fairly brown .
	$-2v$	the hostess was quite impossible in an expensive and very few customers .
Input		**the patrons all looked happy and relaxed .**
AAE	$-v$	the patrons all looked happy and relaxed .
	$-1.5v$	the patrons all just happy and smelled .
	$-2v$	the patrons all just happy and smelled .
DAAE	$-v$	the patrons all helped us were happy and relaxed .
	$-1.5v$	the patrons that all seemed around and left very stressed .
	$-2v$	the patrons actually kept us all looked long and was annoyed .

Table B.8.4: Interpolations between two input sentences generated by AAE and our model on the Yahoo dataset.

Input 1	**what language should i learn to be more competitive in today 's global culture ?**
Input 2	**what languages do you speak ?**
AAE	what language should i learn to be more competitive in today 's global culture ?
	what language should i learn to be more competitive in today 's global culture ?
	what language should you speak ?
	what languages do you speak ?
	what languages do you speak ?
DAAE	what language should i learn to be more competitive in today 's global culture ?
	what language should i learn to be competitive today in arabic 's culture ?
	what languages do you learn to be english culture ?
	what languages do you learn ?
	what languages do you speak ?
Input 1	**i believe angels exist .**
Input 2	**if you were a character from a movie , who would it be and why ?**
AAE	i believe angels exist .
	i believe angels - there was the exist exist .
	i believe in tsunami romeo or <unk> i think would it exist as the world population .
	if you were a character from me in this , would we it be (why !
	if you were a character from a movie , who would it be and why ?
DAAE	i believe angels exist .
	i believe angels exist in the evolution .
	what did <unk> worship by in <unk> universe ?
	if you were your character from a bible , it will be why ?
	if you were a character from a movie , who would it be and why ?

Appendix C

Language Style Transfer from Non-parallel Data

C.1 Proof of Lemma 4

Lemma 4. *Let z be a mixture of Gaussians $p(z) = \sum_{k=1}^{K} \pi_k \mathcal{N}(z; \mu_k, \Sigma_k)$. Assume $K \geq 2$, and there are two different $\Sigma_i \neq \Sigma_j$. Let $\mathcal{Y} = \{(A, b)||A| \neq 0\}$ be all invertible affine transformations, and $p(x|y, z) = \mathcal{N}(x; Az + b, \epsilon^2 I)$, in which ϵ is a noise. Then for all $y \neq y' \in \mathcal{Y}$, $p(x|y)$ and $p(x|y')$ are different distributions.*

Proof.

$$p(x|y = (A, b)) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; A\mu_k + b, A\Sigma_k A^\top + \epsilon^2 I)$$

For different $y = (A, b)$ and $y' = (A', b')$, $p(x|y) = p(x|y')$ entails that for $k = 1, \ldots, K$,

$$\begin{cases} A\mu_k + b = A'\mu_k + b' \\ A\Sigma_k A^\top = A'\Sigma_k A'^\top \end{cases}$$

Since all $\mathcal{Y}$ are invertible,

$$(A^{-1}A')\Sigma_k(A^{-1}A')^\top = \Sigma_k$$

Suppose $\Sigma_k = Q_k D_k Q_k^\top$ is Σ_k's orthogonal diagonalization. If $k = 1$, all solutions for $A^{-1}A'$ have the form:

$$\left\{ QD^{1/2}UD^{-1/2}Q^\top \middle| U \text{ is orthogonal} \right\}$$

However, when $K \geq 2$ and there are two different $\Sigma_i \neq \Sigma_j$, the only solution is $A^{-1}A' = I$, i.e. $A = A'$, and thus $b = b'$.

Therefore, for all $y \neq y'$, $p(x|y) \neq p(x|y')$. □

Appendix D

Language Style Transfer with Confounders

D.1 Classifier Accuracy in the Real Task

Here, we report performance of the invariant classifiers C_s and C_o to classify based on sentiment and category respectively when their coupling is reversed (Section 5.4.2). We take positive reviews from CSJ and HK, and take negative reviews from E and CPA. The test accuracy in shown in Table D.1.1.

Table D.1.1: Classifier accuracy in the real task when sentiment-category coupling is reversed.

Model	**Sentiment**	**Category**
C_{ERM}	64.0	36.0
C_s	**80.4**	-
C_o	-	**62.9**

D.2 Trade-off of Different Weight Combinations of Loss Terms

Table D.2.2: Automatic evaluation results of sentiment transfer from different categories. Accuracies less than 30 are marked in red.

Model	**Sentiment**	**Category**	**PPL**	**BLEU$_{\text{src}}$**
Krishna et al. [66]	22.6	**57.4**	**35.8**	19.2
He et al. [47]	77.7	22.5	44.6	47.6
$\lambda_{C_{\text{ERM}}}, \lambda_{\text{LM}}, \lambda_{\text{BT}} = 1, 1, 1$	87.8	15.8	42.8	48.0
$\lambda_{C_{\text{ERM}}}, \lambda_{\text{LM}}, \lambda_{\text{BT}} = 2, 1, 1$	89.6	14.6	42.4	47.0
$\lambda_{C_{\text{ERM}}}, \lambda_{\text{LM}}, \lambda_{\text{BT}} = 4, 1, 1$	**90.2**	10.3	49.2	47.7
$\lambda_{C_s}, \lambda_{\text{LM}}, \lambda_{\text{BT}} = 1, 1, 1$	72.8	38.7	45.3	61.7
$\lambda_{C_s}, \lambda_{\text{LM}}, \lambda_{\text{BT}} = 2, 1, 1$	78.0	36.4	45.1	59.2
$\lambda_{C_s}, \lambda_{\text{LM}}, \lambda_{\text{BT}} = 4, 1, 1$	85.0	29.5	42.5	53.0
$\lambda_{C_s}, \lambda_{C_o}, \lambda_{\text{LM}}, \lambda_{\text{BT}} = 1, 1, 1, 1$	68.9	56.6	49.8	**63.4**
$\lambda_{C_s}, \lambda_{C_o}, \lambda_{\text{LM}}, \lambda_{\text{BT}} = 2, 1, 1, 1$	73.1	53.3	51.7	61.4
$\lambda_{C_s}, \lambda_{C_o}, \lambda_{\text{LM}}, \lambda_{\text{BT}} = 4, 1, 1, 1$	79.9	50.9	49.2	57.4
Input Copy	3.1	75.3	34.5	100.0

D.3 Additional Examples

Table D.3.3: Additional examples of the synthetic sentiment transfer task.

Input	the new management team is horrible .
Reference	the new management team is great .
Krishna et al.	the new management team is terrible .
He et al.	the new management team is excellent !
M w/ C_{ERM}	the new management team is amazing !
M w/ C_s	the new management team is great !
Ours	the new management team is great .
Input	did n't even get a response .
Reference	even got a response .
Krishna et al.	they didn't even respond to my question .
He et al.	did a great job and the staff is very friendly !
M w/ C_{ERM}	did n't even get a response !
M w/ C_s	did and even get a response !
Ours	always great and always a response .
Input	i got my car detailed here and it looked amazing !
Reference	car was detailed and came out bad !
Krishna et al.	i had a car detailed and it looked incredible !) .
He et al.	i got my car detailed here and it looked awful .
M w/ C_{ERM}	i got my car detailed here and it looked terrible .
M w/ C_s	i got my car detailed here and it looked horrible .
Ours	i got my car detailed here , it looked awful !
Input	they are super friendly , very helpful , and amazingly quick !
Reference	so slow and rude , not helpful at all !
Krishna et al.	they were very friendly , very helpful , and unbelievably quick !) .
He et al.	they are super rude , very helpful , and amazingly slow .
M w/ C_{ERM}	they are super rude , very unprofessional , and amazingly quick .
M w/ C_s	they are extremely rude , not helpful , and amazingly quick .
Ours	they were incredibly rude , not helpful , but amazingly quick !

Table D.3.4: Additional examples of sentiment transfer from different categories.

Clothing, Shoes and Jewelry (negative → positive)	
Input	very thin material . and too short for me . i 'm 5'10 "
Krishna et al.	love it . just too thin , too short , and too short .
He et al.	very nice case . and great for iphone 's . i 'm 5'10 "
Ours	very nice product . and fits fine for me . i 'm 5'10 "
Input	too big for my face
Krishna et al.	too big for my face
He et al.	great fit for my s6
Ours	works great for my face
Home and Kitchen (negative → positive)	
Input	does not solve the lid problem !
Krishna et al.	no way to fix the problem with the lid !
He et al.	does the trick for everything problem !
Ours	does the job the lid problem !
Input	very poor quality . crooked on one end .
Krishna et al.	it is poor quality .
He et al.	very good quality . accurate on one end .
Ours	very good quality . crooked on one end .
Electronics (positive → negative)	
Input	this is great . it easy to use
Krishna et al.	easy to use .
He et al.	this is too small . it had to return
Ours	this was broken . it difficult to use
Input	the drive works as designed
Krishna et al.	the drive works as designed
He et al.	the sized did not work properly
Ours	the drive is too large
Cell Phones and Accessories (positive → negative)	
Input	it is durable and not as bulky as i thought
Krishna et al.	not as heavy duty as i thought it would be .
He et al.	it is uncomfortable and not as bulky as i thought
Ours	it is cheap and not as protective as i thought
Input	excellent quality and perfect fit
Krishna et al.	great quality and perfect fit
He et al.	horrible quality and returned it
Ours	extremely small and tight fit

Bibliography

[1] Ion Androutsopoulos and Prodromos Malakasiotis. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, 38: 135–187, 2010.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[3] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[4] Yannis Assael, Thea Sommerschield, and Jonathan Prag. Restoring ancient text using deep learning: a case study on Greek epigraphy. 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 6368–6375, 2019.

[5] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*, 2018.

[6] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[8] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[9] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[10] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pages 899–907, 2013.

[11] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in neural information processing systems*, 24:2178–2186, 2011.

[12] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Conference on Computational Natural Language Learning*, 2016.

[13] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2): 79–85, 1990.

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

[15] William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. Kermit: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*, 2019.

[16] Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

[17] Benson Chen, Tianxiao Shen, Tommi S Jaakkola, and Regina Barzilay. Learning to make generalizable and diverse predictions for retrosynthesis. *arXiv preprint arXiv:1910.09688*, 2019.

[18] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, 2016.

[19] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

[20] Yo Joong Choe, Jiyeon Ham, and Kyubyong Park. An empirical study of invariant risk minimization. *arXiv preprint arXiv:2004.05007*, 2020.

[21] Ondřej Cífka, Aliaksei Severyn, Enrique Alfonseca, and Katja Filippova. Eval all, trust a few, do wrong to none: Comparing sentence generation models. *arXiv preprint arXiv:1804.07972*, 2018.

[22] Antonia Creswell and Anil Anthony Bharath. Denoising adversarial autoencoders. *IEEE transactions on neural networks and learning systems*, 30(4): 968–984, 2018.

[23] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

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

[25] Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. Avoiding latent variable collapse with generative skip models. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2397–2405. PMLR, 2019.

[26] Chris Donahue, Mina Lee, and Percy Liang. Enabling language models to fill in the blanks. *arXiv preprint arXiv:2005.05339*, 2020.

[27] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. *Advances in Neural Information Processing Systems*, 32, 2019.

[28] Qing Dou and Kevin Knight. Large scale decipherment for out-of-domain machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 266–275. Association for Computational Linguistics, 2012.

[29] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*, 2016.

[30] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018.

[31] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*, 2018.

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

[33] Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and Li Deng. Stylenet: Generating attractive visual captions with styles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3137–3146, 2017.

[34] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[35] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[36] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[37] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Constant-time machine translation with conditional masked language models. *arXiv preprint arXiv:1904.09324*, 2019.

[38] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. 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 6114–6123, 2019.

[39] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[40] Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. Differentiable scheduled sampling for credit assignment. *arXiv preprint arXiv:1704.06970*, 2017.

[41] Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*, 2016.

[42] Tommi Gröndahl and N Asokan. Effective writing style imitation via combinatorial paraphrasing. *arXiv preprint arXiv:1905.13464*, 2019.

[43] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

[44] Jiatao Gu, Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676, 2019.

[45] Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11179–11189, 2019.

[46] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*, 2019.

[47] Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. A probabilistic formulation of unsupervised text style transfer. *arXiv preprint arXiv:2002.03912*, 2020.

[48] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 3, 2017.

[49] R Devon Hjelm, Athul Paul Jacob, Tong Che, Kyunghyun Cho, and Yoshua Bengio. Boundary-seeking generative adversarial networks. *arXiv preprint arXiv:1702.08431*, 2017.

[50] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[51] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Controllable text generation. *arXiv preprint arXiv:1703.00955*, 2017.

[52] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2017.

[53] Daniel Im Jiwoong Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. Denoising criterion for variational auto-encoding framework. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[54] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[55] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[56] Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. Deep learning for text style transfer: A survey. *Computational Linguistics*, 48(1): 155–205, 2022.

[57] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[58] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[59] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.

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

[61] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[62] Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*, 2018.

[63] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[64] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[65] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*, 2017.

[66] Kalpesh Krishna, John Wieting, and Mohit Iyyer. Reformulating unsupervised style transfer as paraphrase generation. *arXiv preprint arXiv:2010.05700*, 2020.

[67] Matt J Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

[68] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

[69] Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*, 2018.

[70] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[71] Juncen Li, Robin Jia, He He, and Percy Liang. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *arXiv preprint arXiv:1804.06437*, 2018.

[72] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer, 2020.

[73] Dayiheng Liu, Jie Fu, Pengfei Liu, and Jiancheng Lv. TIGS: An inference algorithm for text infilling with gradient search. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4146–4156, 2019.

[74] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 469–477, 2016.

[75] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *arXiv preprint arXiv:1703.00848*, 2017.

[76] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[77] Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. Content preserving text generation with attribute controls. In *Advances in Neural Information Processing Systems*, pages 5103–5113, 2018.

[78] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[79] Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabas Poczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W Black, and Shrimai Prabhumoye. Politeness transfer: A tag and generate approach. *arXiv preprint arXiv:2004.14257*, 2020.

[80] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[81] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[82] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[83] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

[84] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[85] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[86] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2013.

[87] Jonas Mueller, David Gifford, and Tommi Jaakkola. Sequence to better sequence: continuous revision of combinatorial structures. In *International Conference on Machine Learning*, pages 2536–2544. PMLR, 2017.

[88] Jonas Mueller, Tommi Jaakkola, and David Gifford. Sequence to better sequence: continuous revision of combinatorial structures. *International Conference on Machine Learning (ICML)*, 2017.

[89] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. 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 188–197, 2019.

[90] Malte Nuhn and Hermann Ney. Decipherment complexity in 1: 1 substitution ciphers. In *ACL (1)*, pages 615–621, 2013.

[91] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[92] Richard Yuanzhe Pang. The daunting task of real-world textual style transfer auto-evaluation. *arXiv preprint arXiv:1910.03747*, 2019.

[93] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[94] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pages 947–1012, 2016.

[95] Ben Poole, Jascha Sohl-Dickstein, and Surya Ganguli. Analyzing noise in autoencoders and deep networks. *arXiv preprint arXiv:1406.1831*, 2014.

[96] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000*, 2018.

[97] Daniel Preotiuc-Pietro, Wei Xu, and Lyle Ungar. Discovering user attribute stylistic differences via paraphrasing. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[98] Reid Pryzant, Richard Diehl Martinez, Nathan Dass, Sadao Kurohashi, Dan Jurafsky, and Diyi Yang. Automatically neutralizing subjective bias in text. In *Proceedings of the aaai conference on artificial intelligence*, volume 34, pages 480–489, 2020.

[99] Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. *arXiv preprint arXiv:2010.05906*, 2020.

[100] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[101] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[102] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[103] Sudha Rao and Joel Tetreault. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. *arXiv preprint arXiv:1803.06535*, 2018.

[104] Paul K Rubenstein, Bernhard Schoelkopf, and Ilya Tolstikhin. On the latent space of wasserstein auto-encoders. *arXiv preprint arXiv:1802.03761*, 2018.

[105] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[106] Cicero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. Fighting offensive language on social media with unsupervised text style transfer. *arXiv preprint arXiv:1805.07685*, 2018.

[107] Anton Maximilian Schäfer and Hans Georg Zimmermann. Recurrent neural networks are universal approximators. In *International Conference on Artificial Neural Networks*, pages 632–640. Springer, 2006.

[108] Allen Schmaltz, Alexander M. Rush, and Stuart Shieber. Word ordering without syntax. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2319–2324. Association for Computational Linguistics, 2016.

[109] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015.

[110] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. *Advances in neural information processing systems*, 30, 2017.

[111] Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. Mixture models for diverse machine translation: Tricks of the trade. In *International conference on machine learning*, pages 5719–5728. PMLR, 2019.

[112] Tianxiao Shen, Jonas Mueller, Regina Barzilay, and Tommi Jaakkola. Educating text autoencoders: Latent representation guidance via denoising. In *International Conference on Machine Learning*, pages 8719–8729. PMLR, 2020.

[113] Tianxiao Shen, Victor Quach, Regina Barzilay, and Tommi Jaakkola. Blank language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5186–5198, 2020.

[114] Tianxiao Shen, Regina Barzilay, and Tommi Jaakkola. Text style transfer with confounders, 2022. URL `https://openreview.net/forum?id=7AzOUBeajwl`.

[115] Yong-Siang Shih, Wei-Cheng Chang, and Yiming Yang. Xl-editor: Post-editing sentences with xlnet. *arXiv preprint arXiv:1910.10479*, 2019.

[116] Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. *arXiv preprint arXiv:1902.03249*, 2019.

[117] Sandeep Subramanian, Guillaume Lample, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. Multiple-attribute text style transfer. *arXiv preprint arXiv:1811.00552*, 2018.

[118] Qing Sun, Stefan Lee, and Dhruv Batra. Bidirectional beam search: Forward-backward inference in neural sequence models for fill-in-the-blank image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6961–6969, 2017.

[119] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[120] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[121] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

[122] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[123] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[124] Rob Voigt, David Jurgens, Vinodkumar Prabhakaran, Dan Jurafsky, and Yulia Tsvetkov. Rtgender: A corpus for studying differential responses to gender. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[125] Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*, 2019.

[126] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[127] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[128] Chen Wu, Xuancheng Ren, Fuli Luo, and Xu Sun. A hierarchical reinforced sequence operation method for unsupervised text style transfer. *arXiv preprint arXiv:1906.01833*, 2019.

[129] Xing Wu, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. Mask and infill: Applying masked language model for sentiment transfer. In *IJCAI*, 2019.

[130] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[131] Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. *arXiv preprint arXiv:1805.05181*, 2018.

[132] Wei Xu, Alan Ritter, William B Dolan, Ralph Grishman, and Colin Cherry. Paraphrasing for style. In *Proceedings of COLING 2012*, pages 2899–2914, 2012.

[133] Yilun Xu, Hao He, Tianxiao Shen, and Tommi S. Jaakkola. Controlling directions orthogonal to a classifier. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=DIjCrlsu6Z`.

[134] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019.

[135] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3881–3890. JMLR. org, 2017.

[136] Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems*, pages 7287–7298, 2018.

[137] Zili Yi, Hao Zhang, Ping Tan Gong, et al. Dualgan: Unsupervised dual learning for image-to-image translation. *arXiv preprint arXiv:1704.02510*, 2017.

[138] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: sequence generative adversarial nets with policy gradient. *arXiv preprint arXiv:1609.05473*, 2016.

[139] Najam Zaidi, Trevor Cohn, and Gholamreza Haffari. Decoding as dynamic programming for recurrent autoregressive models. In *International Conference on Learning Representations*, 2020.

[140] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[141] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. Dialogpt: Large-scale

generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*, 2019.

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

[143] Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander Rush, and Yann LeCun. Adversarially regularized autoencoders. In *International conference on machine learning*, pages 5902–5911. PMLR, 2018.

[144] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.

[145] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

[146] Wanrong Zhu, Zhiting Hu, and Eric Xing. Text infilling. *arXiv preprint arXiv:1901.00158*, 2019.