Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Deep Learning for Text Attribute Transfer on Auto Encoder Models

Bachelor's Thesis

Felix Julius Elias Illes

filles@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Zhao Meng, Damian Pascual Ortiz
Prof. Dr. Roger Wattenhofer

January 4, 2021

Abstract

Natural Language Generation requires models that are able to fully grasp all connotations of the human language in order to generate new meaningful sentences. Current models can effectively capture the grammatical structure and basic content of texts. However, the current state-of-the -art-models are not capable of understanding text specific attributes such as sentiments, politeness, humour and inclinations. This thesis aims to determine whether the current Autoencoder models are suitable for capturing such text attributes. To that end, it will review and compare the performance of current Autoencoder models in terms of Text Style Transfer, the task of modifying a sentences attribute while preserving its content and fluency. We argue that the nature of content and sentiment related information in a text are inherently different. The content relates to what is said, whereas sentiment relates to how the message is conveyed, which implies a multitude of potential nuances. Therefore, understanding and generating sentiment, which is required for Text Style Transfer, is a more complex and intricate task. This thesis analyses the performance of four different Autoencoder models on the task of Text Style Transfer, which is performed by three different adversarial attacks on two datasets. The experiments showed, that all Autoencoder models, except for the VAE model delivered useful results. The SWAE model outperformed the others due to its ability to keep the perplexity low, while still achieving a high Style Transfer Accuracy. We conclude that while Autoencoder models, in general, are capable of capturing sentiments, the models differ in their efficiency of transferring sentiment.

Contents

Chapter 1

Introduction

Deep Learning has become one of the most important areas of Artificial Intelligence and has brought forward impressive results in a multitude of fields. One of these fields is Natural Language Generation where computer processed language is produced on the basis of natural language models. This enables numerous practical applications such as chat bots on a vast variety of online platforms, which respond to customer queries posed in an open chat format. As of now, these models' capabilities are limited to extracting a given input's content, without taking other aspects such as sentiment or politeness into account. One next step in the evolution of natural language models is to add an understanding of specific text attributes relating to sentiments. In order to develop an understanding of text's attributes one commonly uses the method of text style transfer. Joining the field of current efforts, this thesis compares different approaches to text style transfer on multiple models and datasets. To this end, different methods of Text Attribute Transfer and Autoencoders will be reviewed and their effectiveness analyzed.

1.1 Autoencoder Models

The first Autoencoder model (**AE**) was introduced in 1984 [1] by Rumelhart et. al. This model is the eponym for an entire group of models that stem from it. Autoencoders have the ability to capture complex dependencies and to extract a concise, compressed representation. The models' core capabilities are not limited to extracting information. They also have the ability to reconstruct samples. This ability is particularly useful for generating data in a controlled fashion.

1.1.1 General Functionality

Autoencoders encode the input into a condensed latent representation and decode this representation to reconstruct the input while retaining as much of the original information as possible. They make use of two separate networks that are connected by a low dimensional intermediate layer, which is sometimes referred to as the "artificial bottleneck". The first network acts as an encoder transforming the inputs into a low dimensional intermediate representation known as the latent state. The second network, acting as a decoder, takes a vector from the latent state space and re-transforms the information into the original form. The network's weights are then updated by calculating a model-specific loss, whose error terms are propagated back through both networks. This aims to find a condensed data representation in the model's latent space.

1.1.2 Latent Space

The latent space is the representation space for the encoded information. It's size is chosen to be smaller than the input space in order to enforce extraction and condensation of the information into the reduced format. This nudges the model towards finding intricate dependencies which can be stored compactly while still generalizing the information well. Notably, this sometimes leads to obscure dependencies stored in the latent space that are not comprehensible to humans. There are different variations of the Autoencoder model which vary in the way they build and train the latent space. Some variants try to regularize the latent space in order to achieve continuity and completeness. In this context, continuity means that points that are close to each other in the latent space return similar content. Whereas completeness refers to assigning meaningful content to all the points present. These two attributes are required to ensure meaningful results when sampling from the latent representation and enable the models use in a generative setting. The different variants of Autoencoder models differ in their regularization term when building the latent space. As structures the current models were developed and tuned to capture content only, examining the compatibility of current latent space regularization techniques with extracting sentiment would be the natural step forward.

1.2 Text Attribute Transfer

Text Attribute Transfer, also referred to as Text Style transfer, is the task of changing the attribute of a given sentence. For example, one could think of changing the sentiment of a positive sentence such as "The atmosphere in the restaurant was very friendly and openly welcoming." to "The atmosphere in the restaurant was proper and polite." The objective is to maintain fluency and to preserve content while changing the sentiment. Solving this task successfully has implications on a models capability of capturing sentiment and allows for conclusions to be drawn.
Choosing from a plethora of existing approaches this thesis aims to perform Text Style Transfer by modifying the representations in the latent space. This is done

by applying adversarial attacks found in the field of model robustness from the image domain. These attacks aim to fool a classifier, by producing new samples that are similar in content to the original but produce a different label when classified. These existing algorithms are leveraged to perturb the latent space representations of our encoded sentences to match the original in content but produce different labels when classified using a model trained for sentiment. The adversarial attacks used are Deep Fool and Projected Gradient Descent.

Chapter 2

Related Work

Text Attribute Transfer has been studied extensively with a multitude of different approaches. In general, one divides the related work into latent-representation-based and phrase-based approaches. Phrase-based approaches focus on separating content and sentiment phrase by phrase. They directly replace certain phrase, that are detected to convey sentiment and replace them by new phrases conveying the opposite sentiment.[2] Most approaches try to disentangle the latent space by favoring a separate representation of the content and the attribute information. To then rejoin the content vector with a modified attribute vector before decoding.[3][4][5] However, as the sentiment is not only transferred by adjectives, but also by the choice of term for the content i.e. the dictator vs. the president, disentangling is not native and results in sentences of sub-optimal quality.[6]

Unlike these approaches, this thesis follows the previous work conducted by Wang et. al. in their 2019 paper on "Controllable unsupervised text attribute transfer via editing entangled latent representation".[6] The latent space remains entangled, meaning that there is no effort to enforce capturing independent aspects of the information at hand in the latent variables. Text attribute transfer is achieved by perturbing the latent space vectors.

Chapter 3

Encoder and Decoder Cells

The Autoencoder structure is achieved by connecting different neuronal networks in the Autoencoder specific fashion. The two most important neuronal nets are put in the role of the encoder and decoder. These networks can be exchanged without changing the nature of the model, hence they are referred to as cells. This thesis considered two potential candidates for the encoder and decoder networks.

3.1 Transformer cells

The first candidate are transformer based cells as they are used to produce current state of the art text attribute transfer results.[7] It is well known, that training these cells requires a enormous amount of resources and computational capacities. We hypothesized, that using the most common transformer based models as cells is likely to be computationally infeasible. In order to confirm this hypothesis a side experiment was conducted. For more information in this side experiment please see A.

3.2 LSTM cells

The second candidate are "Long Short- Term- Memory"(**LSTM**) networks. They were introduced in 1997 by Hochreiter and Schmidhuber [8] and are based on Recurrent Neuronal Networks (**RNN**). RNNs can be derived from nonlinear ordinary differential delay equations (**DDE**) as can be seen in "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network"[9]. These equations are Partial Differential Equations (**PDE**s) that solve for a variable given its derivative with respect to time. This means that the current value depends on the values observed before. Applying this to text can be done by interpreting text as a discrete time series. The standard RNN cells are unstable when trained, as they suffer from the problem of vanishing gradients. This problem is that, when back-propagating the information over long sequences the information contained in the loss is consumed and converges to 0. LSTM cells address this problem, while still maintaining the same control flow as RNNs. The interior structure of the LSTM-cells is more complex and incorporates multiple mechanisms enabling the flow of gradients.

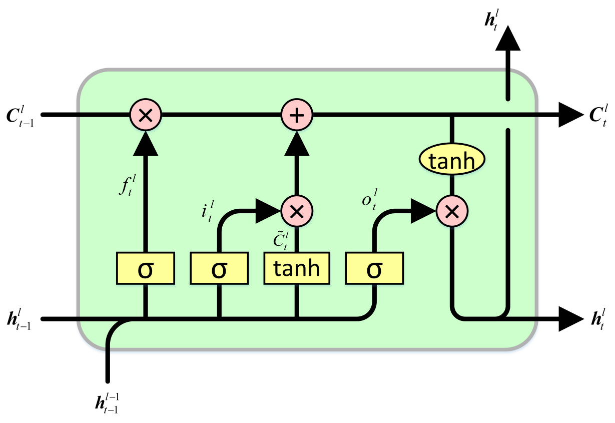

Figure 3.1: Scematic Overview of the LSTM cell

The Graphic 3.1[1] shows that the cell takes 2 inputs, namely a cell state c_{t-1}^l and a hidden state h_{t-1}^l. Another variable used is the previous hidden state h_{t-1}^{l-1}. The cell state is modified via so called "gates" that are controlled by the hidden state input. The gates are either using the *sigmoid* activation, meaning that values get modified in the range from 0 (forgotten) to 1 (kept) or the *tanh* whose output values in the range of $[-1, 1]$.
The first gate is the "forget gate" which modifies the cell state by multiplying the *sigmoid* output calculated by taking the previous cell state and the hidden state input. The second gate is the "input gate" that adds the product of the *sigmoid* and the *tanh* activations of the hidden input and the previous cell state. These updates produce the new cell state. Lastly we have the "output gate" which is used for calculating the next hidden state. This is done by multiplying the *tanh* activation of the new cell state with the *sigmoid* output of the hidden state and previous cell state. This produces the new hidden state.

This LSTM cells are used as encoder and decoder structures in the implementation of this thesis models.

[1]Image was taken from [10]

Chapter 4

Models

Autoencoder models have undergone a somewhat natural evolution. Each model's flaws were addressed by the introduction of the subsequent model. This section aims to reenact these developments while giving detailed insight into each model.

4.1 Autoencoder

The term Autoencoder is ambigous as it refers to the first autoencoder model aswell as to the general class of models.

4.1.1 Definition

The Autoencoder model sometimes referred to as Deterministic or Vanilla Autoencoder was introduced in 1984.[1] The general structure of the model as has been outlined in 1.1.1 is defined by three different entities. We have two different neural nets, one acting as an encoder and the other as an decoder. The third entity is the latent space, which is defined to be smaller in dimension than the other two, creating the artificial bottleneck. These three components are combined to create an Autoencoder model. This model is purely discriminative, having an unregulated latent space. This allows arbitrary distributions in the latent space that are usually placed far from each other in order to optimize for reconstruction. This however makes it impossible to generate sentences in a controlled manner. The models latent space is built in a deterministic fashion.

4.1.2 Loss

First, we need to define CE as the Cross Entropy loss:

$$CE(x_i, y_i) = -\log \frac{\exp x_i[y_i]}{\sum_j \exp x_i[j]} \tag{4.1}$$

The inputs x_i must be a logits vector[1] and y_i is the ground truth label.

$$\mathcal{L}_{rec} = \sum_i CE(log^t_{d,i}, w_{t,i}) \tag{4.2}$$

In this formula the reconstruction loss is defined by taking the cross entropy loss of the decoders output logits (i.e. the probabilities for each word over the entire vocabulary) and the true input.

$$\mathcal{L}_{cla} = \sum_i CE(l_{t,i}, y_i) \tag{4.3}$$

This formula defines the classification loss as defined by taking the Cross Entropy loss between the labels estimated by the classifier and the original labels.

We define a parameter β that is used to emphasise the classification loss when training.

$$\mathcal{L}_{tot} = \mathcal{L}_{rec} + \beta\ \mathcal{L}_{cla} \tag{4.4}$$

The AE models' reconstruction and classification losses are added to generate the total loss used for backpropagation.

4.2 Variational Auto Encoders

Variational Auto Encoders (**VAE**) were introduced by Kingma and Welling in 2014.[11] The main idea behind VAEs is to model the Autoencoders latent space as a continous probability distribution and to regularize it. This is done by introducing a standard multivariate gaussian distribution as a prior. Then a term known as the KL divergence is penalized, taken between the latent (posterior) and the prior distribution. The use of a multivariate gaussian prior is particularly useful, as calculating the KL divergence between two gaussian distributions can be written in a closed form using only the mean vectors and covariance matricies. This prior forces the model to encode the different means around zero and favors non shrinking variances. These introduces non-determinism and allows us to theoretically create infinitely many output sentences.

4.2.1 Definition

VAEs train a latent space that is entirely different from deterministic AEs. For learning a latent distribution the models latent space is split into the latent mean and the latent variance vector. These vectors are used to represent the latent distibution. In order to obtain concrete vectors, necessary for training the decoder

[1]The vector has a length of the number of classes. Each entry in the vector corresponds to the probability that the current sample is of the corresponding class.

samples are drawn. Sampling operations are not compatible with standard back propagation techniques.

4.2.2 Re-parametrisation Trick

In order to train the models using backpropagation the "Re-parametrisation trick" is applied. It is done by moving the stochasticity to a dedicated parameter ε. This allows sampling of a latent vector with the following formula:

$$z = \mu + \sigma * \varepsilon,\ where\ \varepsilon \sim p(x) \tag{4.5}$$

p(x) denotes our prior distribution. This trick allows for the use of back propagation effectively enabling training of the hidden mean vector μ and the hidden variance vector σ.

4.2.3 Loss

The VAE loss is an adapted version of the AE loss. Adapting the reconstruction term to match the use of probability distributions and extending it by adding the Kulback-Leibler (**KL**) divergence term results in the so called Evidence lower bound (**ELBO**) objective.

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q(z|x)}[-log\ p(x|z)] + \gamma\ D_{KL}(q(z|x)\|p(z)) \tag{4.6}$$

Where q(x) denotes the posterior distribution, i.e. the distribution used for approximating our true data distribution. The first term is the reconstruction loss, which is similar to the AE reconstruction term, except the $\mathbb{E}$ operator is used as we are dealing with distributions. The second term is the KL divergence term, which gets scaled by a gamma factor to favour disentanglement of our latent features and is defined as follows.

$$D_{KL}(P\|Q) = \int_X p(x)\ log\frac{q(x)}{p(x)}dx \tag{4.7}$$

Our goal is to penalize a high distance between the assumed prior distribution and the learned latent distribution. The KL divergence provides a tool for assessing the distance between the densities p and q. This is done by measuring the divergence between the two densities point-wise and then summing over the whole domain. This measure is not symmetric and performs poorly in some scenarios. Introducing the KL term to the loss nudges the model towards encoding the data close to each other, providing us with a continuous latent space. This however comes at the cost of a higher reconstruction loss, as small perturbations can result in a change of object reconstructed. These two objective terms are found to be inherently conflicting with each other. Perfect reconstruction would lead to a high KL divergence and a perfect KL divergence implies, that there is no information captured in the posterior distribution leading to a high reconstruction loss. As a consequence a phenomenon known as the posterior collapse occurs.[12] This happens when the model obtains a minimum by minimizing the KL objective to 0. When this happens no information is captured in the posterior and the decoder will not gain any benefits from taking the latent vector into account. It will resort to generating outputs based on the input only in order to lower the reconstruction error, effectively rendering the architecture useless and creating independent x and z distributions. Generating text from such a degenerated latent space is meaningless as there is no information to be used for targeted generation.[13]

Previous work studied the occurrences of posterior collapse and concluded, that it is likely to happen if the decoder is strong enough to lower the reconstruction error by itself or if the latent space does not provide information useful enough. Different solutions have been suggested such as KL-annealing by reducing the encoders stochasticity, dropping the KL penalty or using Word dropout in the decoder in order to weaken the decoders capailities. This as has been thoroughly discussed in the 2019 Paper "Lagging Inference Networks and Posterior Collapse in Variational Autoencoders" by He et. al..[14]

4.3 Wasserstein Auto Encoders

Wasserstein Auto Encoders were introduced by Tolstikhin et. al. in their 2019 paper called "Wasserstein Auto-Encoders".[15] They present a new family of regularized Autoencoders, that keep the good properties of VAEs, including the latent space exhibiting continuity and completeness, as well as the nature of a generative model. Their novelty lies in the regularizer used to build the latent space.

4.3.1 Definition

The solution to posterior collapse was provided by introducing the Wasserstein distance as a new way of regularizing the model. It is based on the Optimal Transport equation. [16] This equation provides us with a measure of distances between two distributions, with a topology that is much weaker than the one defined by the class of f-divergences (of which the KL divergence is a part of). This makes it easier to capture distances in low dimensional space without producing gradients that have a tendency become too large.[15] This regularizer is different from the one used for building the VAE, as it does not require the posterior to be close to the prior in every input sentence x. Instead it imposes constraints on the aggregated posterior of z.

4.3.2 Loss

The Wasserstein loss approximated to train the WAE is defined as follows.

$$\mathcal{L}_{tot} \;=\; \inf_{Q(Z|X)\in\mathcal{Q}} \mathbb{E}_{P_X}\, \mathbb{E}_{Q(Z|X)}[c(X,G(Z))] + \lambda\, \mathcal{D}_Z(Q_Z,P_Z) \tag{4.8}$$

The variables are defined such that $\mathcal{Q}$ is any non-parametric set of probabilistic encoders, P_Z is the prior distribution, P_X is unknown data distribution, P_G defines a latent variable model Q_Z is the distribution of encoded datapoints i.e. $Q_Z := \mathbb{E}_{P_X}[Q(Z|X)]$.[15] We take c as the the reconstruction cost defined in the VAE loss as $\mathbb{E}_{q(z|x)}[-log\ p(x|z)]$. The $\mathcal{D}_Z$ objective of the WAE can be based on two different divergence measures. We base our WAE on the maximum mean discrepancy (**MMD**) which is known to perform well on high dimensional standard normal distributions. It is based on a positive-definite reproducing kernel $k : \mathcal{Z} \ \times \ \mathcal{Z} \to \mathcal{R}$. We choose it to be the inverse multiquadratic kernel commonly chosen for high dimensional Gaussians.[12]

$$k(x,y) = \frac{C}{C + \|x-y\|_2^2} \tag{4.9}$$

$$\mathcal{D} = \| \int_{\mathcal{Z}} k(z,.)dP_Z(z) \ - \ \int_{\mathcal{Z}} k(z,.)dQ_Z(z)\|_{\mathcal{H}_k} \tag{4.10}$$

Where $\mathcal{H}_k$ is defined as the reproducing kernel Hilbert space (**RKHS**), generated by the kernel mentioned above. This Hilbert space is spanned by real-valued functions mapping Z to $\mathcal{R}$.
This provides a good solution to the posterior collapse problem as the loss trains two non-conflicting objectives. However, there is an issue known as stochasticity collapse that can be observed when training WAEs. Namely, that if a distribution with small variance is met with a large gradient coming from a single sample, the Gaussian distribution may degenerate to a Dirac delta function.[12]

4.4 Stochastic Wasserstein Auto Encoder

Stochastic Wasserstein Auto Encoders (**SWAE**) have been proposed by Bahuleyan et. al in their 2019 conference paper "Stochastic Wasserstein Autoencoder for Probabilistic Sentence Generation". [12]

4.4.1 Definition

They provide a solution for the problem of stochasticity collapse. The problem is solved by adding a KL divergence term to each sample. This favors stochasticity, by imposing local priors to each posterior distribution.

4.4.2 Loss

The SWAE loss is defined by adding an extra stochasticity term to the WAE loss.

$$\mathcal{L}_{tot} = \mathcal{L}_{WAE} + \ \lambda_{KL} \sum_{n} KL(\mathcal{N}(\mu_{post}^{(n)}, diag(\sigma_{post}^{(n)})^2) \| N(\mu_{post}^{(n)}, \mathbf{I})) \tag{4.11}$$

Interestingly the KL term is applied sample-wise favoring local stocasticity while avoiding the posterior collapse and the stochasticity collapse.

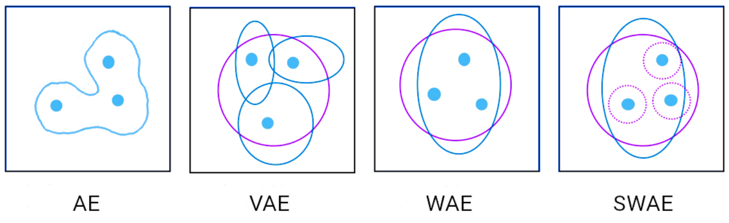

Figure 4.1: Schematic Overview of the Latent Space Regularization

Figure 4.1[2] visualizes the different latent space distributions. The blue lines show the distribution of the latent variable used to calculate the regularization. The purple lines show the prior distribution. The model is set to regularise the latent distributions by producing blue lines i.e. latent distributions that are close to the fixed purple line i.e. the prior. The figure shows that the latent space produced by the basic Autoencoder is not regularized. Therefore the distribution produced is of arbitrary shape and cannot be used for generating new text. For the VAE we can see that the regularization is calculated for each sample individually. This aims to draw all samples to the center ideally producing a continuous and complete latent space. WAE and SWAE use an aggregated latent distribution. Small circles around the dots for SWAE show the regularization added by the stochastic term.

[2]adapted from [4]

Chapter 5

Text Attribute Transfer

In the following a definition and the mathematical formulations for Text Attribute Transfer will be provided. This thesis focuses on doing Text Sentiment transfer by considering two distinct sentiments, positive and negative. Big annotated datasets for other attributes are rare or do not exist. Current Text Sentiment transfer tasks make use of review based data of which there is plenty to be found.

5.1 Definition

As introduced before, text attribute transfer is the task of changing an attribute while maintaining content and linguistic fluency of a given input sentence. Each sentence must be perturbed in such a way, that the new perturbed vector is classified by a label different from the input label. The task can be viewed as an optimisation problem, given by the following formulas. For a set of input sentences $X = \{x_1, ..., x_n\}$ and a set of labels $Y = \{y_1, ..., y_n\}$ the optimization problem is defined as:

$$z = \arg\min_{z^*} \|z^* - E(x)\| \; s.t. \; C(z) = y' \tag{5.1}$$

D denotes the Decoder, E the Encoder, C the Classifier and z denotes the latent vector. This formula can be interpreted as finding the smallest perturbed vector z^* in latent space such that it changes label.
In the following the methods to achieve this objective are discussed.

5.2 DeepFool

DeepFool is an adversarial attack algorithm that was introduced in 2016 by Fawzi et. al.[17] It is used to find the minimal perturbation **r** that is needed to change a sample's estimated label $\hat{k}(\mathbf{x})$.[1]

$$\Delta(\mathbf{x};\hat{k}) := \min_{r} \|\mathbf{r}\|_2 \quad \text{s.t.} \quad \hat{k}(\mathbf{x}+\mathbf{r}) \neq \hat{k}(\mathbf{x}) \tag{5.2}$$

$\mathbf{x}$ denotes a sentence and $\hat{k}(\mathbf{x})$ the corresponding sentiment. Additionally $\Delta(\mathbf{x};\hat{k})$ denotes the robustness of $\hat{k}$ at sample $\mathbf{x}$.
The original paper proposes this attack for quantifying the robustness of deep neural nets used for image classification.[17] In this thesis, we propose applying DeepFool to text data, which is feasible because the perturbations are calculated in a non image domain specific way.
The common principle can be observed when when looking at an affine binary classifier $f(x) = \mathbf{w}^T\mathbf{x} + b$.

$$\mathbf{r}_* := \arg\min \|\mathbf{r}\|_2 \quad \text{subject to} \quad sign(f(\mathbf{x}_0 + \mathbf{r})) \neq sign(f(\mathbf{x}_0)) \tag{5.3}$$

It is not obvious how to enforce the change of the sign.

$$f(\mathbf{x}_0 + \mathbf{r}) = w^T(\mathbf{x}_0 + \mathbf{r}) + b = f(\mathbf{x}_0) + w^T\mathbf{r} \tag{5.4}$$

This equation is used to interpret the change in sign for $f(\mathbf{x}_0 + \mathbf{r})$ in the following way.

$$\mathbf{r}_* := \arg\min \|\mathbf{r}\|_2 \quad \text{subject to} \quad f(\mathbf{x}_0) + w^T\mathbf{r} = 0 \tag{5.5}$$

This can be interpreted as the orthogonal projection of $\mathbf{x}_0$ onto the separating hyper-plane. As the classifier is not necessarily linear, these perturbations are calculated by linearizing the classifier decision boundary locally for each sample $\mathbf{x}_i$.

$$\arg\min_{\mathbf{r}_i} = \|\mathbf{r}_i\|_2 \; \textit{subject to} \; f(\mathbf{x}_i) + \nabla f(\mathbf{x}_i)^T\mathbf{r}_i = 0 \tag{5.6}$$

Where $\mathbf{r}_i$ is the perturbation at iteration step i of the algorithm. This algorithm moves the latent vector towards the decision boundary. This by itself would in many cases not solve the problem as the algorithm is likely to produce perturbations that are close to the decision boundary. These samples are unlikely to exhibit strong sentiment attributes. We expect them to be rather neutral sentences.

5.2.1 Overshoot

To solve this issue Fawzi's Deep Fool paper [17] proposes rescaling the last perturbation vector in order to cross over the decision boundary, This is called the **overshoot** method:

$$\hat{\mathbf{r}}_{over} = (1+\eta) * \hat{\mathbf{r}}$$

[1]This is a paraphrased version of the minimization problem state above.

5.2.2 Constant

Additionally, there is another method for crossing the decision boundary, which we call the **constant** method.

$$\hat{\mathbf{r}}_{const} = \mathbf{c} * \frac{\hat{\mathbf{r}}}{\|\hat{\mathbf{r}}\|_2} \tag{5.7}$$

This allows us to control the distance that we move across the decision boundary more explicitly than the overshoot method. This algorithm can be generalized for multiclass problems by viewing them as multiple One vs. All Classification problems. The generalisation towards nonlinear classifiers works by approximating the decision boundaries as locally linear. Different approaches were compared in section 6.

5.3 Projected Gradient Descent

In addition to the DeepFool method there is the projected gradient descent method as introduced in the 2019 paper titled "Towards Deep Learning Models Resistant to Adversarial Attacks" authored by Aleksander Madry.[18] Projected Gradient Descent (**PGD**) is an adversarial attack based on a initial saddle point formulation:

$$\min_{\theta} \rho(\theta), \text{ where } \rho(\theta) = \mathop{\mathbb{E}}_{(x,y)\ \sim\ D}\left[\max_{\delta \in S} L(\theta, x+\delta, y)\right]$$

Where D defines the underlying data distribution of samples $\mathbf{x}$ and corresponding labels $\mathbf{y}$. In our case the loss function L denotes the Cross Entropy Loss. This lets us interpret the problem in the following way: the inner maximization problem is used to find samples that achieve a high classification loss i.e. a change of label. The outer minimization problem is used to find model parameters robust to the classification.
As our goal is text attribute transfer we lay our focus on the inner maximization problem. This is solved by taking update steps, which are based on the Fast Gradient Sign Methods steps:

$$x + \varepsilon\ sgn(\nabla_x L(\theta, x, y))$$

Taking this step multiple times leads to the **PGD** attack:

$$x^{t+1} = \prod_{x+S} (x^t + \alpha sgn(\nabla_x L(\theta, x, y)))$$

Where S denotes the number of maximum steps and α is used as a scaling factor. We use this new vector to find the minimal perturbation vector z^* as stated in5.1

Chapter 6

Experiments

For this thesis multiple experiments and side experiments were conducted in order to gain further insight into this topic. They were conducted on the clusters of TIK. [19]

6.1 Implementation

The implementation is based on PyTorch. [20] Multiple functions such as the LSTM-cells for the encoder and decoder are implemented using this library. When setting the parameters, we decided to choose parameters similar to [21], [22] and [3] in order to achieve comparability. Our initial learning rate is set to 0.001 and we use Adam for optimization.[23] The hidden, latent and batch sizes are set to 256.

6.1.1 Base Model

The different variants of Autoencoder models are similar in their structure and are built using the same basic operations. These can be summarized to a base model.

Preprocessing

The English language is estimated to entail about 1.000.000 words. Using all of them for our models is not feasible, therefore the data is prepared by sorting the words by frequency and selecting only the most common words. Fortunately the usage of words vaguely follows Zipf's law, stating the the "frequency of use" drops drastically when moving down the ranks. The 9.200 and 58.000 most common words were selected for the Yelp and the Amazon dataset respectively. From the List of chosen words a vocabulary is built using the nltk[24] package, labeling each word with a numbered token. This labeling assigns each word with a unique key known as the word id. In the experiments use of the pretrained glove vectors

[25] is made, as these are used to transfer the words into the 300 dimensional embedding space producing the embedded word vectors. Then the sentence size is fixed to 15 and 20 for the Yelp and Amazon dataset respecively.

Encoding

The embedded sample sentences $\mathbf{x}^t$ are fed to the model's encoder network, producing a cell state $\mathbf{c}_e^t$ and a hidden state $\mathbf{h}_e^t$. (See section 3.2 for detailed explanation of this process.) The artificial bottleneck, which is crucial to Autoencoder models is created by feeding $\mathbf{h}_e^t$ to a linear layer that effectively reduces the dimensions by having a small output size. This layer returns a reduced hidden state vector, which for the basic Autoencoder model translates directly into the latent space vector $\mathbf{l}_t$. For the other models the model specific operations are applied to obtain the latent space vector.

Classification

This thesis approach to Text Attribute Transfer is based on perturbing the latent state vector to move across the decision boundary. Therefore a classifier network is trained to estimate the labels of hidden vectors. This network was chosen as a simple linear layer, returning the predicted probabilities of having a specific label. This probabilities are referred to as classification logits $\mathbf{p}_c^t$. Taking the biggest of these logits returns the predicted label $\hat{\mathbf{y}}$. These are used to train the model for the classification objective and for the perturbation of the latent space vectors.

Decoding

The decoder cell transformes $\mathbf{x}$ and $\mathbf{h}_c^t$ to the decoders cell states $\mathbf{c}_d^t$, the decoders hidden states $\mathbf{h}_d^t$ additionally producing the decoders output logits $\mathbf{log}_d^t$. In contrast to the classification logits, $\mathbf{log}_d^t$'s entries correspond to the probability of an entry being a word in the vocabulary. Therefore, the text can be reconstructed by taking the argmax of $\mathbf{log}_d^t$.

6.2 Datasets

One major issue when doing text style transfer is the lack of annotated datasets. The standard procedure to solve this issue is to take data from different review datasets consisting of text as well as a scoring system of stars. The reviews are divided based on the stars given, into a positve and a negative set.

6.2.1 Stanford Natural Language Inference corpus

We used the Stanford Natural Language Inference corpus (**SNLI**) dataset solely for pretraining the models.[26] The SNLI dataset consists of 629.343 sentences. This dataset was generated by crowd sourcing human workers in an image captioning task and contains short and simple sentences.

6.2.2 Yelp

We used the Yelp Service Reviews dataset which contains 444.101, 63.483, 126.670 samples for the train, validation and test set respectively.[27] The sentences were sourced from yelp customer reviews collected in different large north American cities. We set the maximum sentence length to 15 words. The size of the vocabulary is set to 9.200.

6.2.3 Amazon

Further, we used the Amazon Product Reviews dataset which is larger. It contains 555142, 2000, 2000 samples for train, test and validation set respectively. We trained with a maximum sentence length of 20 words and a vocabulary size of 58.000.

6.3 Training

The training phase is split into 2 separate phases, the pretraining and the training phase. For measuring their results we used the following metrics.

6.3.1 Pretraining metrics

For evaluating the models performance two different metrics were calculated using the reconstructed sentences.

BLEU Score

The **BLEU** [28] Score is a metric commonly used in the field of Natural Language Generation. It is used to measure how well the content of a sentence is kept. It does so by calculating a modification of the n-gram overlap between sentences.

Perplexity

The **Perplexity** score is used to measure how fluent a generated sentence is. This is calculated by feeding the generated sentences to a separately trained language model. That model is then tasked with predicting the next word, for which it returns some probabilities. Perplexity is calculated from the models confidence and the number of potential candidates for each next word. From this we can deduct that a low perplexity score is representative for concise sentences that are similar to what the model knows. For calculating our perplexity scores we used Open-Ai's pretrained GPT-2 model. [29]

6.3.2 Pretraining

We started by pretraining our models on the SNLI dataset, while using a beta of 0 in order to solely focus on lowering the reconstruction bias. This is meant to teach the models the basic understanding of the language without taking the classification into account.

Pretraining SNLI for Yelp								
Models	AE		WAE		SWAE		VAE	
Datasets	test	val	test	val	test	val	test	val
BLEU Score 1	96.86	96.66	97.33	97.29	73.27	73.35	12.36	12.28
BLEU Score 2	95.5	95.2	96.08	96.07	65.03	65.14	3.27	3.31
BLEU Score 3	94.57	94.2	95.19	95.19	59.89	60.03	1.43	1.47
BLEU Score 4	93.71	93.26	94.33	94.35	55.44	55.63	0.68	0.7
Perplexity	31.50	24.04	23.12	23.25	23.25	25.12	-	-

The results were measured on the test and validation sets of the Yelp dataset. When training the VAE model we observed what we believe to be the posterior collapse.

The Amazon dataset is measured with an increased vocab size of 58000 and an increased max length of 20 words. As the maximum number of words used in SNLI, is smaller than 58.000 the vocab size defaults to 36.596.

This thesis aims to compare the performance of the different models and algorithms (Deep Fool constant, Deep Fool overshoot, PGD) set out above. We therefore emphasised comparability over performance. This means that we trained all models with the same set of parameters especially for the same amount of epochs. We observed an insufficient performance on the VAE model, which is most likely

Pretraining SNLI for Amazon								
Models	AE		WAE		SWAE		VAE	
Datasets	test	val	test	val	test	val	test	val
BLEU Score 1	93.93	94.07	93.34	94.49	71.08	71.16	25.56	25.62
BLEU Score 2	91.33	91.52	91.77	91.97	61.82	61.87	9.32	9.3
BLEU Score 3	89.61	89.8	89.96	90.19	56.23	56.25	4.17	4.1
BLEU Score 4	88.03	88.24	88.29	88.54	51.48	51.48	1.69	1.57

due to posterior collapse. This suggestion is supported by the fact, that the encoder and decoder structure are both based on LSTM cells, as discussed above in 4.3. We observed that the SWAE model training takes more epochs for reaching the optimal performance. For comparability we only trained all models for 10 epochs. As a consequence, we estimate that the performance of the SWAE could further be improved by training for more than 10 epochs. Further, it should be noted that all models were trained with a beta of 0.5.

6.3.3 Training

We loaded the pretrained models and trained them on the Yelp dataset. The maximum vocab size of the SNLI dataset is 36.569. The vocab size of the results presented in [21] use a vocab size of 58.000. Loading a model with a smaller vocab size leads to a parameter mismatch. Therefore it is impossible to achieve comparability to the tests conducted for the Yelp dataset as well as to the results presented in previous work[21]. We opted for achieving comparability with previous works and used the vocab size of 58.000. The training results obtained for the Amazon dataset had to be produced by training the models without pretraining them on SNLI.

Training Yelp						
Models	AE		WAE		SWAE	
	test	val	test	val	test	val
BLEU Score 1	95.31	95.8	94.92	95.29	73.18	73.83
BLEU Score 2	93.66	94.26	92.83	93.34	65.85	66.65
BLEU Score 3	92.53	93.21	91.37	91.98	61.33	62.21
BLEU Score 4	91.3	92.05	89.83	90.52	57.22	58.16
Perplexity	31.27	30.50	33.15	30.56	42.65	42.68

Training Amazon						
Models	AE		WAE		SWAE	
	test	val	test	val	test	val
BLEU Score 1	81.47	81.91	85.82	86.14	40.45	40.29
BLEU Score 2	77.05	77.6	81.72	82.15	24.89	24.81
BLEU Score 3	74.65	75.26	79.33	79.83	17.74	17.69
BLEU Score 4	72.77	73.42	77.42	77.96	13.17	13.13

6.4 Text Attribute Transfer

6.4.1 Transfer Metrics

To the five different metrics implemented in "Disentangled Representation Learning for Non-Parallel Text Style Transfer" [21] 2 more were added. The metrics are used to measure the quality of our attacks. When applying an attack to a given input sentence we return a modified version of that sentence, to which we refer as perturbed sentence.

Style Transfer Accuracy

We measured the quality of the algorithms by training a separate classifier that achieves a test accuracy of 0.97637 and validation accuracy of 0.97793 on the Yelp dataset and a test accuracy of 0.71722 and validation accuracy of 0.62499 on the Amazon dataset. We used this separate network to classify the newly generated sentences, using the target labels as ground truth. This provided us with a quantitative measure for evaluating the strength of the transfer. Our model is a simple network consisting of an LSTM encoder and some stacked linear layers using the *relu* and *softmax* activation functions.

Style Transfer Accuracy naive

We measured the quality of attacks by using the models interior classifier as a naive approximation of the Style Transfer Accuracy score. This approximation is biased by the model, and does not provide the same level of objectiveness as an external network does.

Cosine Similarity

As we used the pretrained glove embeddings to build our embedding space, we can safely assume that similar words are located in similar locations in the embedding space. Using this assumption we calculate the Cosine Similarity between the perturbed and the original sentence to get an estimate of content preservation.

Word Overlap

As another measure for content preservation we implemented the Word Overlap score. This score is calculated by taking the unigram word overlap between the original sentence $\mathbf{x}$ and the perturbed sentence $\mathbf{y}$ as $(WO = \frac{count(\mathbf{x} \cap \mathbf{y})}{count(\mathbf{x} \cup \mathbf{y})})$.

Perplexity

We calculate the perplexity in the same way as mentioned in 6.3.1. The only difference being, that we take the mean and the median perplexity while measuring the perplexity of the perturbed sentences.

Geometric Mean

We implemented this measure as proposed in "Disentangled Representation Learning for Non-Parallel Text Style Transfer". [21] It is a measure calculated by taking the geometric mean of the Style transfer accuracy, the Word Overlap and $1/Perplexity$. This is implemented to generate an overall score of the attack, measuring content preservation, transfer strength as well as fluency of the generated sentences in one value.

6.4.2 Deep Fool constant

For these experiments we loaded the trained models and ran the attacks on the dataset's testset. In order to conserve space no legends and axis notations were used. The **x axis** plots the different values for **c** as defined in 5.2.2. **AE** is show in blue, **WAE** is shown in red and **SWAE** is shown in yellow. The experiment

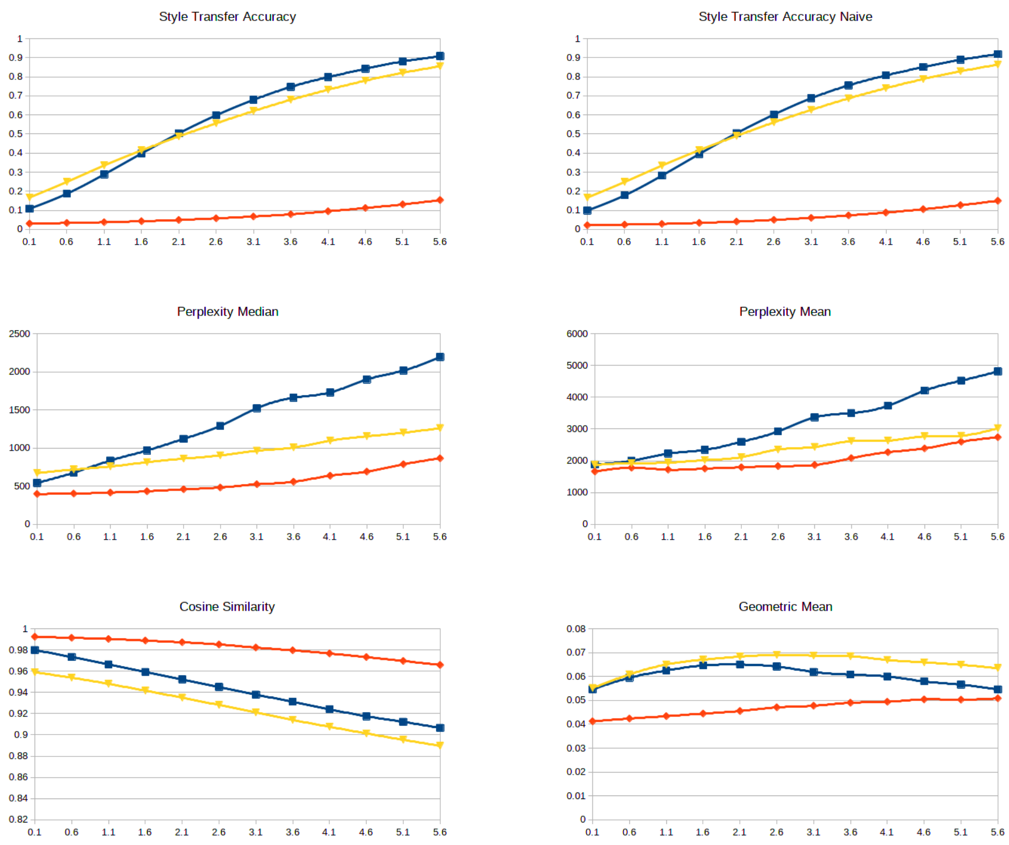

Figure 6.1: Results from Deep Fool constant on Yelp

show promising results. Especially for the AE and SWAE model high Style Transfer Accuracy scores are achieved, meaning that the adversarial attacks are able to cross the decision boundary for almost all samples. For stronger attacks, bigger values of Perplexity are observed, which is similar to what we expected. Interestingly, one can see that for the SWAE model, the increase in Perplexity is much smaller than for the AE model, meaning that its capabilities of transferring fluency are superior. The Geometric Mean which is used to generate an overall score for the attacks shows that the SWAE model outperforms the two other models, and the AE observes a decrease in attack quality with increasing attacks.

6.4.3 Deep Fool overshoot

For these experiments we loaded the trained models and ran the attacks on them on the datasets testset. The **x axis** plots the different values for η as defined in 5.2.1. **AE** is shown in blue, **WAE** is shown in red and **SWAE** is shown in yellow.

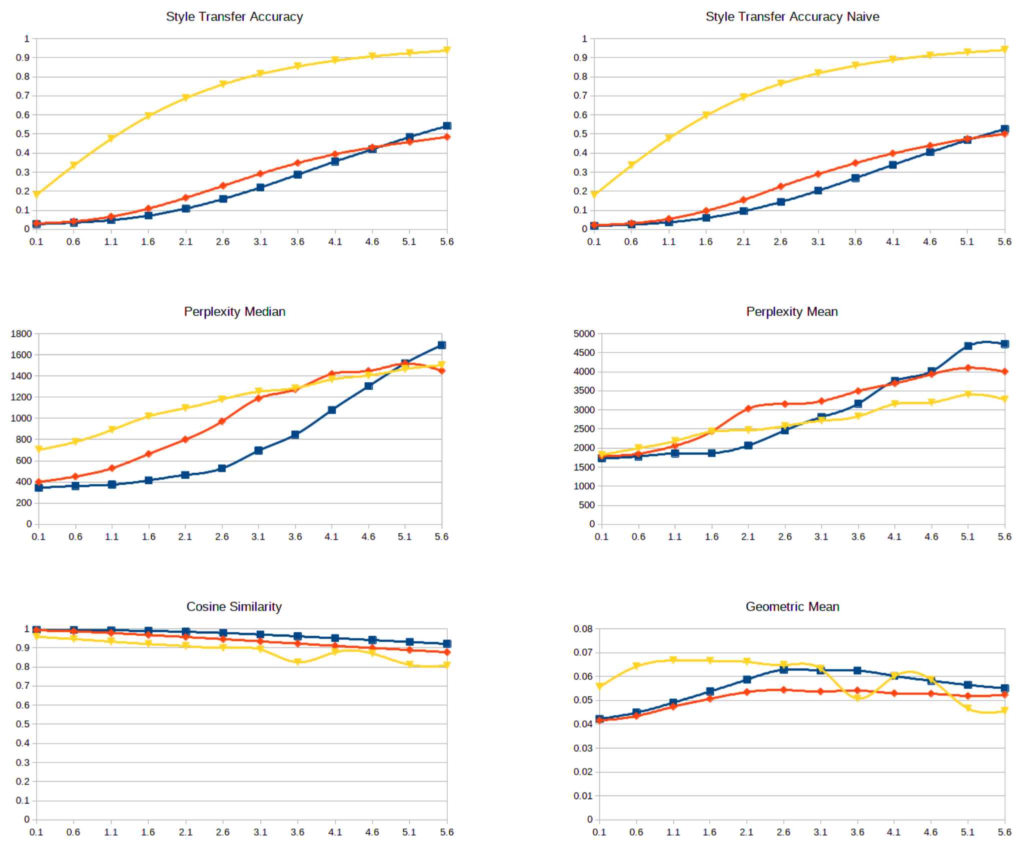

Figure 6.2: Results from Deep Fool overshoot on Yelp

The results obtained show interesting results. Compared to the results from Deep Fool const, we can see, that the WAE is also able to achieve Style Transfer Accuracy results. The AE model performs worse than before, and the SWAE model outperforms the other two. The average attack score as calculated by the geometric mean shows that the overall score stays approximately constant for all three attacks.

6.4.4 PGD

For these experiments we loaded the trained models and ran the attacks on them on the datasets testset. The **x axis** plots the different values for η as defined in 5.2.1. **WAE** is shown in red and **SWAE** is shown in yellow.

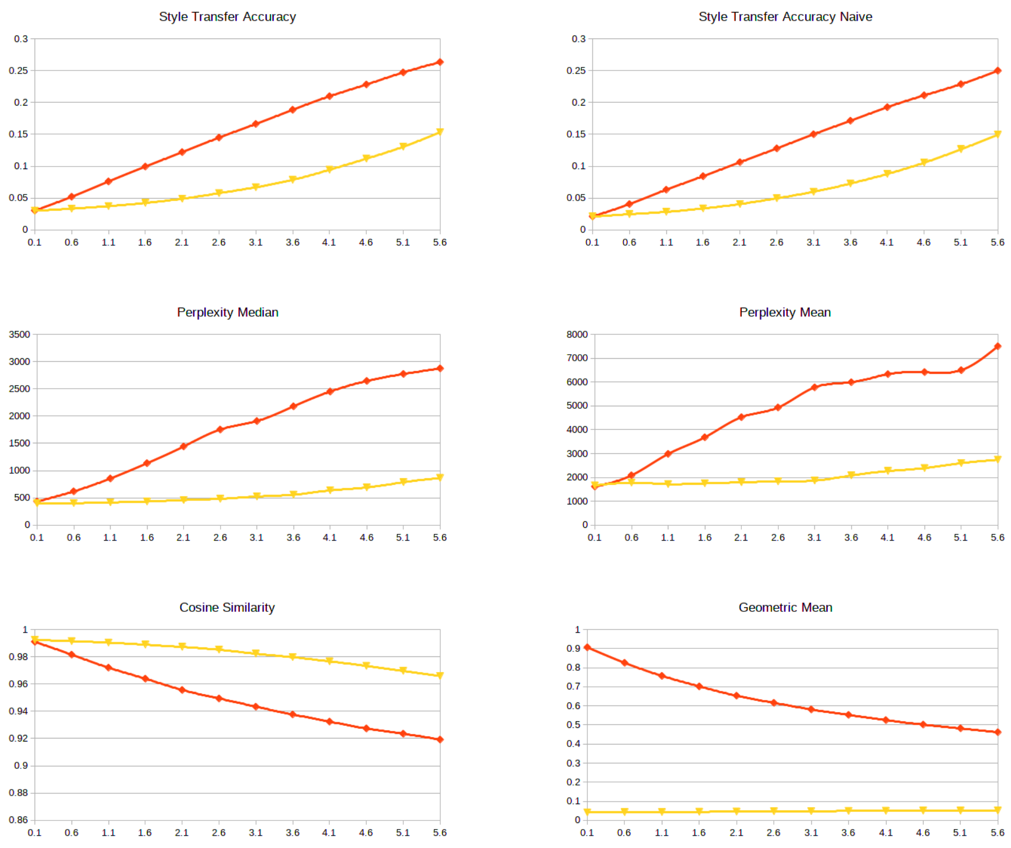

Figure 6.3: Results from Deep Fool overshoot on Yelp

Notably these Attacks achieve only a very small Style Transfer Accuracy of up to 0.3. This result is labeled as insignificant, as these small perturbations can not be utilized for doing Text Style Transfer effiently.

6.5 Latent Space tests

In order to choose the optimal latent space size a side experiment was conducted on the AE model using the Yelp dataset. The goal of this side experiment is to verify our choice for the latent space size. As the thesis focus lies on text attribute transfer, the models need to be able to have a low reconstruction bias even when they are being trained for another objective. Therefore we trained the models using a beta of 1. This emphasises the classification error over the reconstruction error.

Latent Space tests								
Latent Size	32		64		128		256	
SNLI	test	val	test	val	test	val	test	val
BLEU Score 1	69.4	69.76	87.76	87.92	95.34	95.46	97.61	97.61
BLEU Score 2	62	62.4	83.9	84.08	93.65	93.81	96.61	96.59
BLEU Score 3	57.59	58.01	81.54	81.72	92.53	92.72	95.89	95.88
BLEU Score 4	53.74	54.19	81.54	79.62	91.49	91.7	95.2	95.18
Perplexity	41.71	40.64	34.93	35.11	26.62	26.62	23.38	24.40

Latent Space tests								
Latent Size	32		64		128		256	
Yelp	test	val	test	val	test	val	test	val
BLEU Score 1	56.6	57.08	79.92	80.49	92.65	93.07	95	95.35
BLEU Score 2	46.64	47.16	73.8	74.4	90.15	90.64	93.33	93.79
BLEU Score 3	41.08	41.58	70.14	70.82	88.51	89.06	92.18	92.7
BLEU Score 4	36.35	36.81	66.79	67.48	86.83	87.43	90.93	91.49
Perplexity	90.80	87.85	57.57	57.57	38.11	36.19	32.12	32.32

The latent space tests show, that the best BLEU Scores are achieved when using a latent space of 256. Interestingly, we can see that the BLEU scores for a latent space size of 128 are also above 90. These results are similar to what we expected, as larger latent sizes increase the model's capabilities to retain information. Note that values beyond 256 do not make sense, as we embedd the words into a 300 dimensional space. If a value bigger than or equal to 300 were chosen the "artifical bottleneck" that forces the model to extract information would not be present. Interestingly, the scores for latent sizes down to 128 deliver still quite acceptable results. This suggests that the model is able to detect and extract a compressed representation and that complex dependencies can be detected.

Chapter 7

Conclusion

In the following we set out the findings of the experiments conducted and draw conclusions.

7.1 Pre-training and training

The high BLEU scores in the pre-training and training results show, that the models are able to capture the sentences with a low reconstruction bias. The low perplexity scores imply, that reconstructing the sentences is keeping the fluency of the original texts. From this we can conclude that Autoencoders are Natural Language Models that are proficiently able to capture natural language. This paves the way for Text Attribute Transfer.

7.2 Text Attribute Transfer

We undertook the task of Text Attribute Transfer with Autoencoder structures on the basis of 3 different algorithms on two datasets. We hypothesized, that the current state of the art Autoencoder models are able to successfully transfer sentiment. This implies that they are able to capture complex dependencies and extract a meaningful representation of sentiment.

7.2.1 Results

The plots for the Style Transfer Accuracy, allow us to conclude, that the labels are changed effectively when perfroming DeepFool. In particular, this holds true for the AE and the SWAE model which achieve high scores in both variants of the DeepFool attack.

As anticipated, we observed an increased perplexity, but the SWAE model managed the growth of data properly. As we kept our latent space entangled and do not differentiate between sentiment and content information, the drops in cosine similarity were also similar to what we expected. The measurement that indicates the overall efficiency of a given attack is the geometric mean. We observed that the geometric mean is approximately constant, which supports the hypothesis that the attacks can be carried out effectively.

Finally, we deduct from the results obtained, that, in principle, the models are capable of successfully performing Text Attribute Transfer. As the success of the task is tied to the model‘s capability to capture sentiment, we can conclude that our experiments confirm our hypothesis that the current state-of-the-art Autoencoder models‘ structures can be used to capture sentiment. Moreover, we can conclude from our experiments that the SWAE model performs the best even under the circumstance of having a higher reconstruction bias than the other models.

7.3 Latent Space

The Latent Space Experiments were conducted by measuring the performance of the autoencoding task when using multiple latent sizes.

7.3.1 Results

The latent space test was conducted in order to determine the optimal latent space size for conducting the text attribute transfer task. The results have shown that a latent size of close to the embedding size produce the best results. This is congruent with our expectations. Larger latent space sizes lead to an increase in the model‘s capacities of transferring information. It is interesting to see, that smaller latent space sizes of down to 128 achieve considerable BLEU scores. This implies that the model is able to detect and extract meaningful information, that can be represented in a condensed format. We hypothesized that text data exhibits complex structures and dependencies, that can be extracted by the model. This is backed by the results.

Bibliography

[1] H. G. . W. R. Rumelhart, D., "Learning representations by back-propagating errors," in *Nature 323, 533–536*, Jul. 1986.

[2] J. Li, R. Jia, H. He, and P. Liang, "Delete, retrieve, generate: A simple approach to sentiment and style transfer," 2018.

[3] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," 2018.

[4] G. Lample, S. Subramanian, E. M. Smith, L. Denoyer, M. Ranzato, and Y.-L. Boureau, "Multiple-attribute text rewriting," in *ICLR*, 2019.

[5] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola, "Style transfer from non-parallel text by cross-alignment," 2017.

[6] K. Wang, H. Hua, and X. Wan, "Controllable unsupervised text attribute transfer via editing entangled latent representation," 2019.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[9] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," p. 132306, Mar 2020.

[10] dec 2020. [Online]. Available: https://i.stack.imgur.com/RHNrZ.jpg

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014.

[12] H. Bahuleyan, L. Mou, H. Zhou, and O. Vechtomova, "Stochastic Wasserstein autoencoder for probabilistic sentence generation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4068–4076. [Online]. Available: https://www.aclweb.org/anthology/N19-1411

[13] A. Goyal, A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio, "Z-forcing: Training stochastic recurrent networks," 2017.

[14] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick, "Lagging inference networks and posterior collapse in variational autoencoders," 2019.

[15] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," 2019.

[16] L. Ambrosio, *Lecture Notes on Optimal Transport Problems.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 1–52. [Online]. Available: https://doi.org/10.1007/978-3-540-39189-0_1

[17] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," 2016.

[18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2019.

[19] ""https://computing.ee.ethz.ch/services/slurmhardware"."

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[21] V. John, L. Mou, H. Bahuleyan, and O. Vechtomova, "Disentangled representation learning for non-parallel text style transfer," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.* Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 424–434. [Online]. Available: https://www.aclweb.org/anthology/P19-1041

[22] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola, "Style transfer from non-parallel text by cross-alignment," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 6830–6841. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/2d2c8394e31101a261abf1784302bf75-Paper.pdf

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[24] S. Bird, "Nltk: The natural language toolkit," in *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Lan-*

guage Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics, 2002.

[25] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[26] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[27] N. Asghar, "Yelp dataset challenge: Review rating prediction," 05 2016.

[28] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, "Bleu: a method for automatic evaluation of machine translation," 10 2002.

[29] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[31] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[32] C. Li, X. Gao, Y. Li, B. Peng, X. Li, Y. Zhang, and J. Gao, "Optimus: Organizing sentences via pre-trained modeling of a latent space," 2020.

Appendix A

Experiments

A.1 Amazon Attacks

The attacks on the amazon dataset were conducted in the same experimental setting as the attacks on the yelp dataset. Running these attacks led to an CUDA ERROR stating an overflow of memory in the GPUS. This made measuring the perplexity scores of training, as well as generating data for the adversarial attacks impossible. This issue was reproduced with down scaling the batch size from 256 to 128 and 64. As these experiments were very time consuming (training one model on the amazon dataset takes up to 18 hours for batch size 64.) and the training results for amazon are deemed unpromising, no further experiments were conducted on that matter.

A side experiment as mentioned in 3.1 on our transformer based auto encoder was conducted proving it computationally infeasible for our purpose.

A.2 Transformer cells

The model structure from the 2019 paper: Controllable Unsupervised Text Attribute Transfer via Editing Entangled Latent Representation [6] was reproduced by using what we believe to be the most popular transformer based models as cells. BERT[30] was used as an encoder network and GPT-2 [31] was used as a decoder network. The result was found to be similar in style to the Optimus model.[32]
Our experiments show, that this setup is computationally infeasible. When training, we observed that the model's need for memory capacity exceeds the capacity available on the Arton Clusters[19]. The batch size was lowered to 16 instead of 256 which produced the same result. For further investigation the BERT model was downscaled to use only: 6 hidden layers (default:12), 6 attention heads(default: 12), 15 max position embeddings (default:512), 9.200 vocab size (default: 30.522), 256 hidden size (default:768) producing the same result. The GPT-2 network was downscaled in a similar fashion still exceeding memory ca-

pacities. As a reference we show, the parameters used in [6] For encoder and decoder they use two stacked layers of transformer cells, which use 1024 dimensional Feed-Forward Networks, a hidden size of GRU of 128 and batch size 128. They set their embedding size, latent size and dimension size to 256.

From this experiment we conclude that using BERT and GPT-2 is not computationally feasible for doing Text Attribute Transfer when using our implementation of the base model and our training loop.