Ludwig-Maximilians-Universität München
Faculty for Mathematics, Computer Science and Statistics

Master Thesis

Anonymization of Text Data with Attention-Based Networks

Author:

Lina Teresa Marie Fischer

Supervisor:

Prof. Dr. Christian Heumann
Matthias Assenmacher

30th January 2020

Abstract

Anonymization of Text Data with Attention-Based Networks

The present master thesis tries to combine a concept of automatic data anonymization with a mechanism from the field of neural networks. Since a large part of daily communication contains very personal data, the proposed approach is tested on a large email corpus. It is the objective to automatically replace any sensitive data allowing re-identification by, at best, thematically close words and leave other data unchanged. The success of such an approach could save both work and privacy concerns in the era of artificial intelligence. A basic knowledge of statistics and machine learning is required to understand this thesis. The necessary basics concerning the mathematical concepts of privacy and the field of neural networks are explained extensively within the thesis.

Contents

1 Introduction

Data security is a very current topic. Since the introduction of the General Data Protection Regulation (GDPR) in May 2018, there have been new legal regulations on the handling of personal data. In GDPR, personal data are defined as all data that can be attributed to a (natural) person [(20), Ch.1, Art.4]. Of course, the risk to disclose personal data is minimized if all personal data is deleted or not even collected but this approach is incomplete. Its incompleteness lies in that it ignores the accompanying loss of valuable, non-personal data and also the frequent necessity of data collection for research. An example of this is medical research, which could greatly benefit from the use of patient records or the area of product development that depends on user experience. However in the case of Amazon's Alexa, the unfiltered collection of this data has so far posed an unmanageable security risk, as it does not even only the device listen but also Amazon employees [(40)]. Even if technological and political progress manages to prevent unintended eavesdropping, there lies a general risk in sending personal data like one's position to a corporate server. Any servers can be a target for hackers, who can conveniently find an aggregation of sensitive data there. As a countermeasure, the GDPR proposes *privacy by design*. Privacy by design means to include a privacy mechanism within the technology to prevent, in respect of the purpose, unnecessary, personal data to be saved in the first place [(20), ch.4, Art.25]. Obviously, there is no simple solution as for instance not transmitting the position may disable Alexa to respond to a request. The central dilemma in data security gets obvious:

How can data be private while still contain core information?

One solution could be the usage of anonymized data. Anonymization techniques reaching a trade-off between both objectives even hold further advantages. For instance, when it comes to image recognition like damage detection on cars. If the mechanism learned which car belongs to whom instead of the examination of accident tracks it would not be extendable to images of different people's cars. If anonymized data were used the mechanism would be forced to focus on the individual damage instead of individual car holders.
Besides the advantages in terms of generalization, intelligent solutions in the area of anonymization are beneficial for several parties: they allow the partial use of confidential data for research purposes while at the same time they protect the data security of the individual. The topic of this master thesis is the handling of the conflict between the usability of data and the degree of anonymization incorporating two different mathematical areas: The concepts of **differential privacy** as intelligent anonymization definition and **self-attention**, a recent development in the area of neural networks enabling a network to learn to focus on different parts of the input data at different points in time.

2 Methods and related approaches

2.1 Privacy and Automation

2.1.1 Earlier Privacy Approaches

The privacy concept used within the application part of this master thesis is so-called **differential privacy**. However, before its introduction there were several simpler concepts of anonymization. Three of them will be described below. All of them try to reach a **trade-off between privacy and data utility** which constitutes the central challenge in the area of data anonymization. The reason therefore lies in the fact that in general, the utility declines when the level of privacy rises. Assume for example that a study on the efficacy of administered drugs in autoimmune diseases is to be conducted at a clinic. In order to create a representative study, a large number of patient records are required. In contrast, autoimmune diseases are very rare. Trying to collect new information for the study would be very expensive on the one hand and very inefficient on the other hand, since the information required already exists in the form of medical records at many different medical institutions. However, these files cannot simply be passed on for reasons of confidentiality. Instead, at least all specific information such as name, place of residence, age and sex should be concealed. This is essential, as there are already studies that show that people can be identified directly by the three attributes gender, date of birth and postal code [(42)]. But it is precisely these attributes that can also provide valuable information about the effectiveness of a drug. For example, side effects affecting the circulation certainly correlate with age. The place of residence can also provide information about special dietary requirements or location specific diseases. Simply deleting this data may lead to a loss of information and therefore data utility. In contrast, the mere mitigation of the informative degree of the data (for example, using intervals “between 75 and 85” instead of “80 years old”) may preserve those information but not sufficiently conceal the identity and hence lack an adequate level of privacy. As the level of privacy is to be **maximized** while the decrease in data utility shall be **minimized** the general problem can be formalised the following statement:

$$\min_{\text{utility loss}} \max_{\text{degree of privacy}}$$

But in order to optimize the levels of privacy and utility, a way to measure both of them must be determined first. This section explains anonymization strategies built on mathematical definitions with hyperparameters to be tuned.

The following descriptions of k-anonymity, l-diversity and t-closeness are all based on [(9)].

k-anonymity

The idea of k-anonymity is to make amendments to each observation in a way such that it ”is indistinguishable from at least $k-1$ other records“ [(9), p.2]. So, it tries to offer privacy by hiding individuality in homogeneous groups of at least k members. These groups result from ”generalization“ or ”suppression“ of attributes that serve to identify individuals [(9), p.2]. k is serving as the hyperparameter regulating the trade-off between privacy and utility. The more adapted records are linked to one original record, the more privacy is introduced. The sentence

”John Age 80 has Alzheimer“

is used in [(9)] to illustrate measures and is adopted here also. In order to apply suppression, the name "John" is then replaced by 'NA', but randomly choosing a different name from "John" but used in the database would have worked just as well. The diagnosis "Alzheimer" must also be replaced and can be generalised with the term "dementia". The resulting sentence then reads

"'NA' Age 80 has dementia".

Whether further attributes should be replaced depends on the reached indistinguishability. If there are $k-1$ other people of Age 80 with a dementia diagnosis they *protect* each other with their identical appearance as there are k *original* records equally linked to the anonymized record. Unfortunately, it is emphasized that "adversaries that use background knowledge" and data sets with quite similar row entries represent exceptions where k-anonymity failed to prevent re-identification [(9), p.2]. This is rather conclusive as in a database where only John has dementia not much information is hidden.

l-diversity

l-diversity extends k-anonymity in a way such that a large enough variety of sensitive attributes is provided. The recipe is to ensure that "there are at least l "well-represented" values for each confidential attribute"[(9), p.2]. In this technique, the number of required possible replacements l corresponds to the hyperparameter - the more variety, the more privacy but the less potential utility. In the case that there are not l different replacements, it is necessary to invent new replacements. But this has a negative influence on the data quality as the underlying data structure is violated. Taking the example from above, and wishing to replace the name "John" with a different name, either a re-identification is facilitated if there are only two possible replacements or inconsistencies may arise as there are now l names instead of 3. An example for such an inconsistency may be that several diagnoses for one patient could be torn into several diagnoses for several patients which is confounding the facts. Hence, l-diversity does not solve k-anonymity's drawbacks completely. Additionally, the introduced bias and possibly too high similarity among the outcome records are said to counteract the usability of the approach. Thus there is also room for improvement in this technique.

t-closeness

The violation of the underlying data structure when applying l-diversity to a data set is found to be reflected best in missing "semantic closeness"[(9), p.2]. On the hypothesis that semantic proximity can be measured by the proximity of the distributions of the previous and adjusted attributes, t-closeness proposes to limit the distance between the distributions to t. t then corresponds to the hyperparameter regulating the trade-off between privacy and utility. The higher the discrepancy in distributions the higher the privacy but the lower the utility in consequence. In comparison to k-anonymity and l-diversity, t-closeness is able to produce an anonymous dataset. However, it is stated that "sometimes it is at the expense of the correlations"[(9), p.2], as the forced distribution will change some relations. While this may disguise personal information it may also influence the data utility.

All three methods are said to either lack strategies to find optimal values or have so far been described as excessively computationally expensive. Below, the concept of differential privacy is explained. This method discards high computational effort and is automated more easily,

thereby tackling some of the problems faced by the methods above. Still, the choice of suitable hyperparameters is left to the applicant.

2.1.2 Differential Privacy

All descriptions concerning differential privacy are based on [(1)] and [(13)].

The main concept of differential privacy is to change the outcome of a trained model by adding noise in a fashion such that removing single individuals from the training data does no longer affect the outcome remarkably. In other words, the goal is to prevent any re-identification by hiding all individuality. Instead, the likeliness of any result shall be influenced by a tiny factor only. In the following, D and D' are datasets whose L_1-distance is less than 1 ($||D - D'||_1 < 1$).

Definition. ($(\epsilon$-$\delta)$-Differential Privacy) [(1), p.17]
For ϵ, $\delta \geq 0$ and any two data sets D and D' a function (mechanism) $\mathcal{M} : D \to \mathrm{R}^d$ mapping a dataset D to a probability fulfills $(\epsilon - \delta)$-differential privacy if

$$\mathbb{P}(\mathcal{M}(D) \in S) \leq e^{\epsilon} \cdot \mathbb{P}(\mathcal{M}(D') \in S) + \delta \tag{1}$$

for all $S \subseteq \mathrm{Range}(\mathcal{M})$.

Obviously, for $(\epsilon, \delta) = 0$, both $\mathbb{P}(\mathcal{M}(D) \in S) \leq \mathbb{P}(\mathcal{M}(D') \in S)$ and $\mathbb{P}(\mathcal{M}(D') \in S) \leq \mathbb{P}(\mathcal{M}(D) \in S)$ are valid and it follows that

$$\mathbb{P}(\mathcal{M}(D) \in S) = \mathbb{P}(\mathcal{M}(D') \in S). \tag{2}$$

So the more similar D and D' are, the more indistinguishable outcome produces the differentially private mechanism.

Remark 1. (Privacy Budget) [(13), p.4]
The value pair (ϵ, δ) controls the amount of privacy and is therefore referred to as **Privacy Budget**. The pair corresponds to the hyperparameters in differential privacy.

In general, δ is often discarded and only ϵ is used to adjust the degree of privacy. As a consequence, a mechanism is called ϵ-differential private then.

Remark 2. [(13), p.4]
Depending on whether the privacy budget is applied on the total or a specific subset of all individuals, there is a distinction between **global** and **personalized privacy budget**.

Differential privacy is achieved with the application of an (ϵ, δ)-differentially-private mechanism. Here, two mechanisms are specifically relevant: the **Laplace Mechanism** and the **Gaussian Mechanism**. Both are described in terms of numeric queries $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ together with their according privacy guarantees. All descriptions concerning the Laplace mechanism follow [(1), p.31-32] if not indicated differently.

Definition. (Laplace Mechanism)
For an arbitrary $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, the Laplace Mechanism is defined as

$$\mathcal{M}_{\text{Lap}}(x, f(\cdot), \epsilon) = f(x) + (Y_1, ...Y_k), \tag{3}$$

where $Y_i \overset{\text{i.i.d}}{\sim} \text{Lap}(\frac{\Delta f}{\epsilon})$, $i \in \{1, ..., k\}$.

Here, Δf corresponds to the **L_1-sensitivity** of f:

$$\Delta f = \max_{x,y \in \mathbb{N}^{|\mathcal{X}|}, ||x-y||_1=1} ||f(x) - f(y)||_1. \tag{4}$$

Furthermore, a random variable X is distributed according to a zero-centered **Laplace probability density** with scale parameter b if

$$X \sim \text{Lap}(x|b) = \frac{1}{2b} \exp(-\frac{|x|}{b}). \tag{5}$$

Theorem 1.
The Laplace mechanism is differentially private with privacy budget $(\epsilon, 0)$.

The corresponding proof is also provided here for better understanding.

Proof. Let $x, y \in \mathbb{N}^{|\mathcal{X}|}$ be such that $||x - y||_1 \leq 1$. Let p_x and p_y denote the probability density induced by $M_{\text{Lap}}(x, f(\cdot), \epsilon)$ respectively $M_{\text{Lap}}(y, f(\cdot), \epsilon)$. Then, for $z \in \mathbb{R}^k$ aribtrary:

$$\begin{aligned}
\frac{M_{\text{Lap}}(x, f(\cdot), \epsilon)(z)}{M_{\text{Lap}}(y, f(\cdot), \epsilon)(z)} \sim \frac{p_x(z)}{p_y(z)} &= \prod_{i=1}^{k} \frac{\exp -\frac{\epsilon|f(x)_i - z_i|}{\Delta f}}{\exp -\frac{\epsilon|f(y)_i - z_i|}{\Delta f}} \\
&= \prod_{i=1}^{k} \exp\left(\frac{\epsilon(|f(y)_i - z_i| - |f(x)_i - z_i|)}{\Delta f}\right) \\
&\overset{\text{triangle ineq.}}{\leq} \prod_{i=1}^{k} \exp\left(\frac{-\epsilon(|f(x)_i - f(y)_i|)}{\Delta f}\right) \\
&= \exp\left(\frac{\epsilon||f(x) - f(y)||_1}{\Delta f}\right) \leq \exp(\epsilon).
\end{aligned} \tag{6}$$

□

In the third step, the so-called triangle equality could be applied as $|\cdot|$ represents a metric. Because it will be needed for further understanding, a short revision is provided here in accordance with [(41)].
A **metric** evaluates distances and refers to a measure d incorporating three attributes:

1. $d(x, x) = 0$ and $d(x, y) = 0 \implies x = y$ (self-reflexivity),
2. $d(x, y) = d(y, x)$ (symmetry) and
3. $d(x, y) + d(y, z) \geq d(x, z)$ (triangle equality).

In the case that $d(x, y) = 0$ does not necessarily indicate $x = y$, d does not denote a metric but a **pseudo-metric** [(13), p.4].

The Gaussian mechanism works in a similar way, but as the name suggests a Gaussian noise variable is added instead of a Laplace variable.
All information concerning the Gaussian Mechanism is taken from [(1), p.53]. In contrast to the L_1-sensitivity used in the Laplace mechanism, the L_2-sensitivity constitutes a part in the definition of a differential-private Gaussian mechanism. It is defined as

$$\Delta_2 f = \max_{x,y \in \mathbb{N}^{|\mathcal{X}|},\, ||x-y||=1} ||f(x) - f(y)||_2, \tag{7}$$

where $|| \cdot ||_2$ indicates the corresponding 2-norm, usually the euclidean norm.

Definition. (Gaussian Mechanism)
The following mechanism

$$\mathcal{M}_{\text{Gaussian}}(x, f(\cdot), \sigma) = f(x) + (Y_1, ..., Y_k), \tag{8}$$

where each $Y_i \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \sigma^2)$, for $i \in \{1, ..., k\}$, is called **Gaussian Mechanism**.

Theorem 2. For any $\epsilon \in (0, 1)$, the Gaussian noise mechanism is (ϵ, δ)-private if

$$c^2 > 2 \ln \frac{1.25}{\delta} \tag{9}$$

and

$$\sigma \geq c \cdot \frac{\Delta_2(f)}{\epsilon}. \tag{10}$$

The according proof follows a similar structure than the above proof and is therefore discarded here.

Both mechanisms provide the understanding that differential privacy works because a zero-centered noise distribution is added to the true data generating distribution p_{data} and consequently blurring the boundaries between different classes like user names. The formulation in terms of numerical queries refers to one of the most common types of data in need of anonymization, namely tables containing integers. Nonetheless, the definitions may be extended to continuous data in an obvious way but in settings involving unstructured data different definitions may be of use.

2.1.3 (Paper) Generalised Differential Privacy for Text Document Processing

While the above descriptions are advantageous in that they are formulated in sort of a general way and rather easily computed in comparison to preceding alternatives, the anonymization of text data holds some characteristics that require adjustment. In (21), an approach of *author obfuscation* on a task published on the platform PAN@Clef is described. As the anonymization algorithm within the application section of this thesis is mainly based on this paper, the additional privacy definitions are presented here along with the approach.

For the input text, a so-called "bag-of-words representation" (bow) is used. This representation reduces the input text but does interestingly neither prevent author re-identification nor topic classification. A **bag-of-words** representation of a document simply lists the unique words used together with their respective frequency while other attributes as for instance the order are discarded. Taking the following example sentence

„Deep down in the ocean, small fish clean the mouths of large fish. "

the corresponding bag of words looks as follows:

{„Deep": 1,
„down": 1,
„in": 1,
„the": 2,
„ocean": 1,
„small": 1,
„fish": 2,
„clean": 1,
„mouths": 1,
„of": 1,
„large": 1}.

Here, the article „the" is used twice but is not likely to reveal relevant information. This applies to all so-called **stopwords**, mostly „prepositions, pronouns and articles" [(21),p.5], and hence they are often discarded when it comes to bag-of-words or text representations in general. In the approach presented, the researchers also deleted stopwords during construction of the bags. The ultimate goal of the paper is to

„provide a mechanism which changes the input without disturbing its topic classification, but that the author can no longer be identified" for „a(n) given (...) input bag-of-words representation of a text document". [(21), p.2]

Therefore, a privacy framework for unstructured data as text is necessary to evaluate amendments made to the original data. A modification of differential privacy, so-called $d_{\mathcal{X}}$**-privacy** is explained first. It is then further extended to contain a measure of topic-relatedness of two bags of words b and b' and finally used to find a mechanism $\mathcal{M}$ transforming b and b' into differentially private representations $\mathcal{M}(b)$, $\mathcal{M}(b')$.

First, the extension of differential privacy is described. While the former formulation of differential privacy refers in a rather general way to sets of data points, a different formulation is advantageous for unstructured data as text. With $d_{\mathcal{X}}$-privacy „a metric-based extension of differential privacy " [(21), p.2] was developed.
In order to be able to apply mathematical operations on words or characters, a word vector representation *Vec* is trained first. A representation of this kind is called **word embedding** and will be explained in a later section. At this point it is sufficient to know, that there simply is a distinct vector representation for each unique word and a document can be represented by the set of all vectors corresponding to the words contained.

In the paper recited, distances between words w_1, w_2 are evaluated by comparing the distances between their representation vectors $\mathrm{Vec}(w_1)$, $\mathrm{Vec}(w_2)$:

$$d_{\mathrm{Vec}}(w_1, w_2) := d(\mathrm{Vec}(w_1), \mathrm{Vec}(w_2)). \tag{11}$$

Consequently, if the distance measure d fulfills the necessary conditions of a pseudo-metric, so does d_{Vec}. It must be emphasized that measuring semantic resemblance with spatial proximity is only useful if the word vector space indeed reflects relations between words. As explained in a later chapter, word embeddings like Word2Vec, FastText and GloVe indeed manage to capture semantics.

In comparison to differential privacy, $d_{\mathcal{X}}$-privacy incorporates the idea that the degree of privacy mainly depends on the spatial distance between a data set and its anonymized equivalent. Again, it is therefore required that the embedding reflects semantic properties. Below, $\mathbb{B}\mathcal{X}$ describes the set of all bags-of-words and $\mathbb{D}(\mathbb{B}\mathcal{X})$ the set of possible probability distributions over the bag-of-words space. As previously emphasized, the application of a differentially private mechanism corresponds to the rearrangement of an input in relation to a distribution. Hence, every privacy mechanism $\mathcal{M}$ maps $b \in \mathbb{B}\mathcal{X}$ to the probability space $\mathbb{D}(\mathbb{B}\mathcal{X})$: $\mathcal{M} : \mathbb{B}\mathcal{X} \to \mathbb{D}(\mathbb{B}\mathcal{X})$. In the following, $\mathcal{M}(b)(Z)$ describes the probability that amendments to b made by the mechanism $\mathcal{M}$ result in a representation contained in the subset $Z \subseteq \mathbb{B}\mathcal{X}$. With a valid choice of metric or pseudo-metric as $d_{\mathcal{X}}$, $d_{\mathcal{X}}$-privacy can be defined. The below definition is derived from [(32), p.3] as the definition was not contained in [(13)] and was furthermore adapted to the case.

Definition. ($d_{\mathcal{X}}$-privacy)
Let $d_{\mathcal{X}}$ define a (pseudo-)metric on the vector space containing the word embedding. A mechanism $\mathcal{M} : \mathbb{B}\mathcal{X} \to \mathbb{D}(\mathbb{B}\mathcal{X})$ is called $\epsilon d_{\mathcal{X}}$-private if

$$\mathcal{M}(b)(Z) \leq e^{\epsilon d_{\mathcal{X}}(b,b')} \mathcal{M}(b')(Z) \tag{12}$$

for any bags of words b, b' and any subset $Z \subseteq \mathbb{B}\mathcal{X}$.

Obviously it would be beneficial to determine the distance of the topics of two documents. E.g., the **Earth Mover's Metric** measures the difference in topics of two inputs. Its definition as the solution to a transport optimization problem is presented below.

Definition. (Earth-Mover's Distance)
For two bag-of-words representations X and Y and a (pseudo-)metric d_S on the set of finite bags S, the **Earth Mover's Distance (EMD)** solves

$$E_{d_S}(X, Y) := \min_F \sum_{x_i \in X} \sum_{y_j \in Y} d_S(x_i, y_j) F_{ij} \tag{13}$$

under the following constraints:

1. $\sum_{i=1}^{n} F_{ij} = \frac{b_j}{|Y|}$
2. $\sum_{j=1}^{l} F_{ij} = \frac{a_i}{|X|}$, $F_{ij} \geq 0, 1 \leq i \leq k$, $1 \leq j \leq l$.

Here, a bag of words X (Y) is of the form

$$X = \{x_1^{a_1}, x_2^{a_2}, ..., x_k^{a_k}\} \ \ (\text{resp. } Y = \{y_1^{b_1}, y_2^{b_2}, ..., y_l^{b_l}\}),$$

where there are k different words in X and l different words in Y. F_{ij} corresponds to an entry in a so-called **flow matrix** $F \in \mathbb{R}_{\geq 0}^{k \times l}$ indicating „the (non-negative) amount of flow from $x_i \in X$ to $y_j \in Y$"[(21), p.4]. A flow-matrix loosely refers to how many words in X flow to one word in Y in order to transform X into Y. Its values are usually part of the linear optimization problem whose solution is given by the Earth Mover's Distance. However, in the case that $|X| = |Y|$, it can be shown that

$$F_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j. \end{cases} \tag{14}$$

For $|X| = |Y|$ the Earth Mover's Distance corresponds to a pseudometric. This pseudometric is called **Earth Mover's Metric**.

Example.
To illustrate how the EMD is calculated, three bags of size 5 are compared to each other. Let F be the optimal flow matrix

$$F = \begin{pmatrix} \frac{1}{5} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{5} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{5} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{5} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{5} \end{pmatrix} \tag{15}$$

For three sentences the difference in topic shall be evaluated.

a) „This year the winter in France begins early."

b) „In 2019, snow fell in Italy before Christmas."

c) „On the continent, melted snow is also called water ."

After stopword removal, the three sentences constitute the following bags-of-words:

a) $b_1 = \{$year, winter, France, begins, early$\}$

b) $b_2 = \{$2019, snow, fell, Italy, Christmas$\}$

c) $b_3 = \{$continent, melted, snow, called, water$\}$

To be able, to calculate vector distances, a vector representation must be found for all trained words first. For the sake of simplicity, the word vectors of Google's Word2Vec model are used. When setting d to be the Euclidean metric

$$||x_i - y_i||_2 = \sqrt{\sum_{1 \leq k \leq n} \left(x_i^{(k)} - y_i^{(k)}\right)^2}, \tag{16}$$

the distance in topic between b_1 and b_2 is calculated as

$$E_d(b_1, b_2) = \frac{1}{5} \cdot (d(\text{year}, 2019) + d(\text{winter}, \text{snow}) + d(\text{France}, \text{fell}) + d(\text{begins}, \text{Italy}) + d(\text{early}, \text{Christmas})).$$

Using the pretrained Word2Vec Embedding with 300−dimensional vectors, the Earth-Mover's-Metric between the bags of words corresponds to $E_d(b_1, b_2) = 3.3296$, $E_d(b_1, b_3) = 3.6710$ and $E_d(b_2, b_3) = 4.0060$. The Earth-Mover's Metric therefore supports the intuitive decision that b_1 and b_2 are closer related in topic.
Now, the calculation of the Earth Mover's Metric is shown but the adaptation of differential privacy to a privacy term incorporating topic-relatedness is still outstanding. The incorporation of topic-relatedness is desirable because of two reasons. On the one hand, remaining topic-relatedness corresponds to preserved data utility in terms of content similarity. On the other hand, distance in topic-relatedness causes masking of the authorship and consequently a certain level of privacy. Consequently, this term of privacy already incorporates a trade-off between privacy and utility. In the below definition, the (pseudo-)metric $d_{\mathcal{X}}$ is simply replaced with the Earth Mover's Metric $E_{d_{\mathcal{X}}}$ induced by $d_{\mathcal{X}}$.

Definition. (Earth-Mover's Privacy)
For a data set $\mathcal{X}$, a (pseudo-)metric $d_{\mathcal{X}}$ on $\mathcal{X}$, a mechanism $\mathcal{M} : \mathbb{B}\mathcal{X} \to \mathbb{D}(\mathbb{B}\mathcal{X})$ is called $\epsilon E_{d_{\mathcal{X}}}$-private iff

$$\mathcal{M}(b)(Z) \leq e^{\epsilon E_{d_{\mathcal{X}}}(b,b')} \mathcal{M}(b')(Z) \tag{17}$$

for all b, $b' \in \mathbb{B}\mathcal{X}$ and $Z \subseteq \mathbb{B}\mathcal{X}$. $E_{d_{\mathcal{X}}}$ refers to the Earth Mover's Metric with $d_{\mathcal{X}}$ as distance metric.

Up to this point, no $d_{\mathcal{X}}$-private mechanism incorporating a privacy-related measure is provided. Its derivation is part of the application chapter. A different approach describing the adaptation of differential privacy to text data is explained as part of the embedding section. As the training of embedding spaces depends on neural networks, their architecture is explained before.

2.2 Deep Learning with Neural Networks

A basic introduction to machine learning is presumed in order to follow this master thesis. Nevertheless, some of the main ideas are briefly recapitulated in accordance with [(7), ch.5].
One of the main challenges in machine learning is to make sure that a good balance between model complexity and generalization error is found when selecting the model. Either the model may underfit and not be able to represent the training data properly or it is trained in a way too focused on the training data and overfit as it does not generalize well to previously unknown test data. The trade-off is mainly controlled over the model's **capacity** which indicates the model's ability to represent complex relationships. The number of trainable parameters indicates a degree of complexity, as the model is the more flexible the more parameters there are. However, this flexibility also leads to a more exact adaptation to the training data and consequently the model produces worse results for previously unseen observations. The typical setup is illustrated in the figure below.

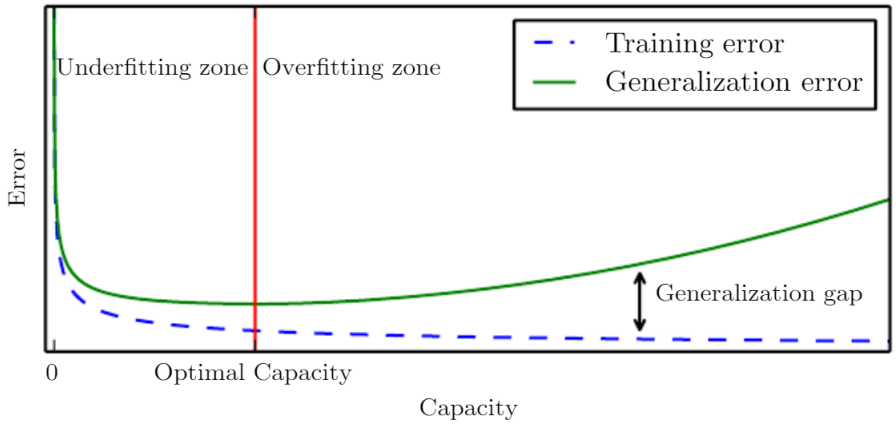


Figure 1: „Typical Relationship Between Capacity and Error“ [(7), ch.5, p.113]

Deep learning is a subarea of machine learning setting new standards regarding the number of parameters used and therefor inheriting the susceptibility to overfitting and underfitting. As in machine learning, deep learning also tries to address this problem by the application of regularization techniques. And also in other respects the strategy for finding the best model corresponds to the usual procedure. First, a suitable model class must be chosen, then a loss function has to be selected that adequately reflects the objectives of the learning process and finally optimization methods for loss minimization and consequently parameter optimization must be picked out.
Now follows a description of the motivation for the use of the model class of neural networks, as well as of its functionality and fundamentally important (or especially for this master thesis important) concepts concerning optimization. The subsequent explanations regarding the area of deep learning are all based on [(7), ch.6] and [(8)] if not indicated differently.

The term **deep learning** refers to all machine learning methods involving neural networks. While most other typical machine learning methods are able to tackle tasks requiring linear feature transformations only, its necessity lies in the strength of finding non-linear representations. XOR states a simple example of a classification problem in need of a non-linear decision boundary.

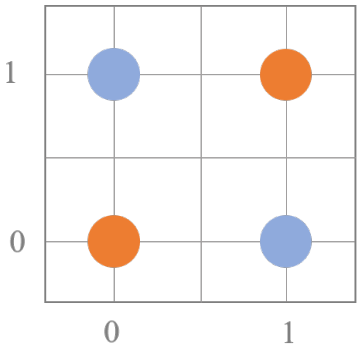


Figure 2: The XOR Problem Statement (inspired by [(8), ch.1]

The input space X consists of four different observations:

$$X = \{(0,0),(0,1),(1,0),(1,1)\} \tag{18}$$

The figure shows that obviously no linear decision boundary is able to distinguish between the orange and blue samples without the systematical misclassification of one observation. The correct classification looks as follows:

(x_1, x_2)	y
$(0,0)$	False
$(0,1)$	True
$(1,0)$	True
$(1,1)$	False

The labels must be encoded to numbers and y is binary encoded here: $y = (0,1,1,0)^T$. Obviously, a transformation of the input space is needed. The following small neural network is able to perform this transformation.

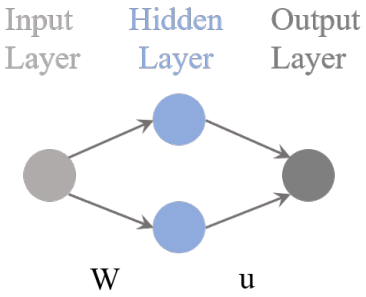


Figure 3: Neural Network for XOR Problem Statement (inspired by [(8), ch.1])

The figure shows the general structure of neural networks consisting of **input neurons**, **hidden neurons** and **output neurons**. Another common term for neuron is **cell**. Neurons are not randomly put in the same layer but all neurons in the same layer are computed in parallel. While one hidden layer is sufficient to model the XOR problem, usually there are several hidden layers. A possible model structure containing several hidden layers is displayed below.

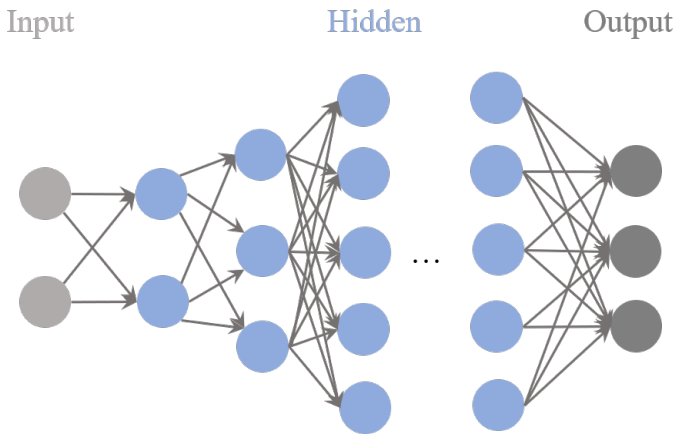


Figure 4: Neural Network with Several Hidden Layers and Multiple Output Neurons (Own Illustration)

Several output neurons are sometimes necessary, for example if it comes to multiclass classification. Between all layers there are directed connections which are basically the product of the previous neuron's output and a corresponding weight. Afterwards, a so-called **activation function** is applied to the weighted value. The output value of each hidden neuron in the i-th layer is calculated as [(34), p.227]

$$z_j^{(i)} = f\left(\sum_{k=1}^{M} w_{kj}^{(i)} \cdot z_k^{(i-1)} + w_{k0}^{(i)}\right) \tag{19}$$

where f indicates the activation function inside the cell, z_k^{i-1} refers to the output of the k-th neuron in the previous ($(i-1)$-th) layer and $w_{kj}^{(i)}$ denotes the weighting of $z_k^{(i-1)}$ in the j-th neuron of the $i-th$ layer. $w_{kj}^{(i)}$ hence describes the weight along the connection from the k-th neuron in layer $i-1$ to the j-th neuron in layer i. With $w_{k0}^{(i)}$ the corresponding bias in the i-th layer is denoted. The following illustration aims to visualize the notation for a network involving a bias term in each linear projection.

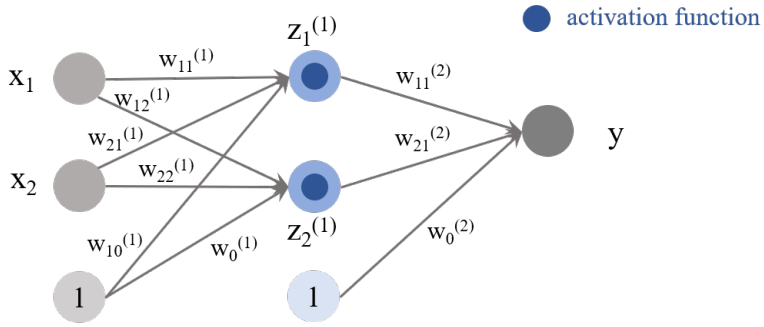


Figure 5: Neural Network with Notations of Cells and Connections (inspired by [(8), ch.1])

In the above formula, the previous hidden layer consists of M neurons, hence $k \in \{1, ..., M\}$. The choice of the applied activation function is arbitrary as long as the function is differentiable and non-linear. Nevertheless, those functions should be elected with regard to the desired output. The subsequent section about activation functions is additionally oriented in [(30), p.13-14].

In general, activation functions can be divided into **threshold functions** and **sigmoidal functions**. Threshold functions are functions applying different operations to values below and above a threshold. A very common activation function of this kind is

$$\text{ReLU}(z) = \max(z, 0), \tag{20}$$

the **RE**ctified **L**inear **U**nit. Activation functions are applied not only inside hidden layers but also in the output layer and therefore they determine the output type. Threshold functions are a good choice if the model outputs scores and no continuous transformation is needed. In case that the output values are preferred to be squashed in between a range $[a, b]$ in a continuous way, a sigmoidal function is the activation function of choice. All sigmoidal functions ψ are monotonous and

$$\lim_{z \to -\infty} \psi(z) = a \text{ and } \lim_{z \to \infty} \psi(z) = b. \tag{21}$$

Due to this restriction, such functions inherit an S-shaped curve. The most prominent example of a sigmoid is the **logistic sigmoid**:

$$\psi(z) = \frac{1}{1 + \exp(-\alpha z)}. \tag{22}$$

Here, α is a control parameter with effect on the slope. The logistic sigmoid outputs values between 0 and 1 and is therefore suited to be interpreted as a probability. Its choice is common in binary-classification tasks, however in multiclass classification use of the **softmax** activation is preferable:

$$\text{softmax}(z)_i^{(k)} = \frac{\exp(z_i^{(k)})}{\sum\limits_j \exp(z_j^{(k)})} \tag{23}$$

Within softmax activation, the value of one unit z_i is related to the values of all units in the same layer. Another example for a frequently used sigmoid is the **hyperbolic tangent function**

$$\psi(z) = \tanh z. \tag{24}$$

This function maps values in $[-1, 1]$ insted of $[0, 1]$.
In the XOR example, the input consists of two features and hence there are two input neurons. ReLU is chosen as activation function in the hidden layer and no activation function (respectively $id(z) = z$ as activation function) is selected for the output layer of the XOR model. In fact, the classification problem is now modeled with a regression model as there is a linear output unit and not a sigmoid.

Let

$$W^{(1)} = \begin{pmatrix} w_{11}^{(1)} & w_{21}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, W_0^{(1)} = \begin{pmatrix} 0 & -1 \\ 0 & -1 \\ 0 & -1 \\ 0 & -1 \end{pmatrix}, W^{(2)} = \begin{pmatrix} 1 \\ -2 \end{pmatrix} \text{ and } W_0^{(2)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

then

$$z_{\text{pred}} = \max(XW^{(1)} + W_0^{(1)}, 0)W^{(2)} + W_0^{(2)} \tag{25}$$

This is a chained function whose depth depends on the number of hidden layers. A neural network with n hidden layers can be evaluated as a chain of $n+1$ activation functions, each applied on a weighted sum of inputs from the previous layer. All weights are part of the optimization problem but must be initialized. The matrix notation of the weights enables easy parallel computation for all observations in X:

$$\max\left(\underbrace{\begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}}_{X} \underbrace{\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}}_{W} + \underbrace{\begin{pmatrix} 0 & -1 \\ 0 & -1 \\ 0 & -1 \\ 0 & -1 \end{pmatrix}}_{W_0}, 0\right) \cdot \underbrace{\begin{pmatrix} 1 \\ -2 \end{pmatrix}}_{W^{(2)}} + \underbrace{0}_{W_0^{(2)}} = \tag{26}$$

$$\max\left(\begin{pmatrix}0&0\\1&1\\1&1\\2&2\end{pmatrix}+\begin{pmatrix}0&-1\\0&-1\\0&-1\\0&-1\end{pmatrix},0\right)\cdot\begin{pmatrix}1\\-2\end{pmatrix}=\begin{pmatrix}\max(0,0)&\max(-1,0)\\\max(1,0)&\max(0,0)\\\max(1,0)&\max(0,0)\\\max(2,0)&\max(1,0)\end{pmatrix}\cdot\begin{pmatrix}1\\-2\end{pmatrix}= \tag{27}$$
$$\begin{pmatrix}0&0\\1&0\\1&0\\2&1\end{pmatrix}\cdot\begin{pmatrix}1\\-2\end{pmatrix}=\begin{pmatrix}0\\1\\1\\0\end{pmatrix}, \text{ which meets exactly the solution to the XOR problem.}$$

A small neural network was able to solve an earlier as intractable described issue. The property as a **universal approximator** is illustrated here and represents one of the greatest advantages of neural networks. More precisely, the **Universal Approximation Theorem** states that a neural network with only one hidden layer is in fact capable of expressing any function. There are still limitations to this universality, as for many functions the suitable network structure contains a huge number of parameters or the model is not necessarily able to learn the corresponding parameters. But the theorem certainly shows that neural networks technically enable the representation of far more complex models.

2.2.1 Input Shapes

The first layer of a neural network is always the input layer whose number of neurons corresponds to the input dimensionality. This seems quite straight-forward if the inputs are the four-dimensional vectors from the XOR example. However, images, videos, text or audio data have a different dimension, which must also be considered in the input layer. If, for example, an image is to be displayed numerically, this initially results intuitively in a matrix displaying pixels as entries. Consequently, the matrix dimension corresponds to *length* $\times$ *width*. But this matrix lacks any information about the color of the individual pixels. An additional axis is necessary to bring the color of the pixels into play. Each pixel and thus each matrix entry then gets an additional dimension with the corresponding RGB values. A color image of size 64×64 has then the shape $(64, 64, 3)$ and contains therefore $64 \times 64 \times 3 = 12228$ values. In addition, usually a **batch** of samples is input, as in the XOR example when several vector samples were read in (combined into a matrix). A fourth dimension is necessary then and the resulting input shape will correspond to (batch_size $, 64, 64, 3$).
In the application part of this thesis, text data is processed only. The representation of text data depends on the encoding of its features which are usually words or n-grams (collections of n characters). A simple way is the use of **one-hot-encoding** if there is a low number of unique words (n-grams) only. In one-hot-encoding, each of n words (n-grams) is assigned an n-dimensional unit vector. However, if there is a larger vocabulary, the use of a trained embedding is recommended for text. Embeddings of this kind will be described later in detail as part of the section about embeddings. At this point it is sufficient to know that in an embedding all features are represented by vectors and that the vector size is usually much smaller than the cardinality of the vocabulary. So, when it comes to text, the input shape is determined by the *batch_size*, the number of features and lastly the size of the embedding vector. The figure below aims to illustrate how to imagine a word as a vector.

Figure 6: Encoding Words as Vectors (Own Illustration, **Left**: Trained Embedding, **Right**: One-Hot-Encoding)

The representation of a sentence (or a document or another cohesive list of words) as a whole is the collection of all vector representations of the contained words.

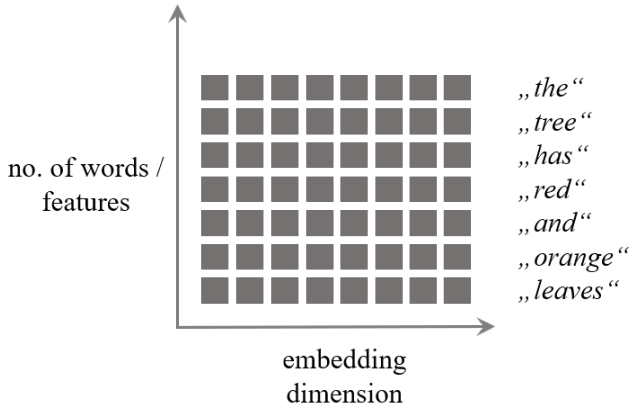


Figure 7: Matrix Representation of a Sentence (Own Illustration)

Finally, the input batch consists of several sentences (a sequence of matrices) that constitute one tensor.

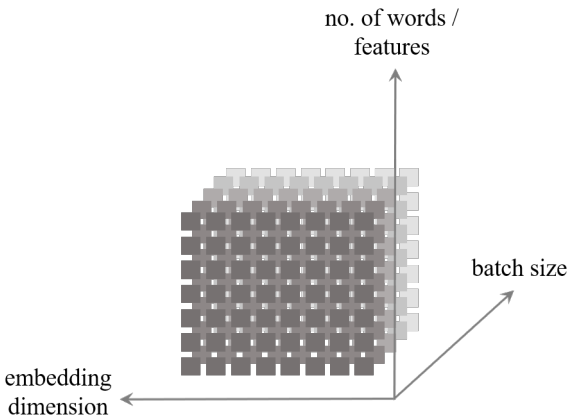


Figure 8: Text Input as Tensor (Own Illustration)

Whether text or not, any categorical output must also be encoded. Again, one-hot-encoding represents a possibility as well as any different less dimensional but distinct representation.

2.2.2 Optimization

All information in this section is based on [(4)], [(6), ch.9], [(7), ch.8] and [(8), ch.2].
A neural network can only solve a specific task if its weights have been optimized specifically for the problem at hand. If one has already decided to present the problem with the help of a network and thus answered the question of representation, it is necessary to find a suitable loss function. This loss function should punish misstatements of the network during the optimization

process and reward correct predictions. Since the ultimate goal is to model a probability distribution for given observations as realistically as possible, the neural network approach can also be described as an attempt to estimate the true data generating distribution p_{data}. In statistics, the parameters of a density function selected for a problem are determined using given observations. This approach is referred to as **Maximum Likelihood Estimation (MLE)**, however its revision is no part of this thesis. However, from the MLE the preference for optimizing the concave log-likelihood is taken. In terms of a cost or loss function, the (convex) **negative log-likelihood** reflects well the requirements for the optimal parameters, since it assumes very large values for observations considered unlikely or impossible under the presumed data generating process p_{data}:

$$p_{\text{data}}(x|\gamma) \to 0 \qquad \Longrightarrow \qquad -l(x) = -\log p_{\text{data}}(x|\gamma) \to \infty \tag{28}$$

In addition, during the optimization process, those parameters γ should be favored which are as likely as possible with respect to the given observations $x \in \mathcal{X}$. And indeed, the negative log-likelihood function $-l(x)$ assumes the lowest cost for these parameters.
However, the model shall not be evaluated for one observation x only. The expected cost caused by applying the model f on all given observations compared to the true p_{data} measures the quality of model parameters in a comprehensible manner. This value is referred to as risk $\mathcal{R}(f)$. For instance, in a supervised setting where the model f aims to simulate the joint distribution of observed data points $x \in X$ and corresponding observed labels $y \in Y$ based on a training set of observed (x, y)-pairs, the risk $\mathcal{R}$ incorporating the negative log likelihood $-l(x)$ as loss function $L(f(X), Y)$ looks as follows:

$$\mathcal{R}(f) = \mathbb{E}_{x \sim p_{\text{data}}} L(f(X), Y) = -\mathbb{E}_{x \sim p_{\text{data}}} \log(p_{\text{data}}(x|\gamma)) \tag{29}$$

This function also measures the so-called **cross-entropy loss** between a model and the true data distribution p_{data}. The term has its origin in information theory and corresponds to evaluating how much information is lost when a distribution is "encoded" with another distribution.
If p_{data} was known, the attempt to model the distribution would not consist of a learning procedure taking training samples as input and optimizing weights in sort of a trial-and-error manner. It is the specifics of machine learning challenges that there is no knowledge of the underlying distribution. Consequently, the risk minimization results in **empirical risk minimization** and the empirical mean is evaluated instead (implicitly evaluating the mean according to the empirical data distribution $\hat{p}_{\text{data}}$):

$$\mathcal{R}_{\text{emp}}(f) = \frac{1}{|X|} \sum_{x \in X} L(y|f(x)) \tag{30}$$

The loss functions' attempt to minimize the distance between the true underlying data generating process p_{data} and the distribution simulated by the model is crucial. Distance measures as $\mathbf{L_1}$ and $\mathbf{L_2}$, respectively the **Mean Squared Error (MSE)** or **Mean Absolute Error (MAE)** are common choices in regression tasks while the cross-entropy loss is the most popular loss in classification tasks. Their choice is no coincidence, all of them can be traced back to the optimization of a negative log-likelihood loss.

There are more loss functions and the decision should be taken anew for each task, on the one hand with respect to the learning objective and on the other hand with respect to properties in terms of optimization. Also, the negative log-likelihood is a rather exact method which therefore increases the risk of overfitting. In this respect, **surrogate loss functions** which hold a similar objective but discard some precision represent a precaution.
An iterative optimization approach is necessary in order to minimize the empirical risk. In this respect, gradient descent methods are attractive procedures and described next.

The Gradient Descent Method

For an arbitrary, differentiable and unrestricted function f a minimum x_{opt} can be determined via a *descent method*. A descent method is characterized by

$$f(x_{k+1}) < f(x_k) \text{ for all } k < k_{\text{opt}}$$

and equality is fulfilled for $k \geq k_{\text{opt}}$. The *Gradient descent algorithm* represents a prominent way to use the descent information derivation provides to find a minimum. The idea behind gradient based optimization can easily be illustrated with the help of a wanderer lost in the mountains. In desperation, the hiker wants to find the valley (the global minimum of the mountain's surroundings) only by the fastest route. He therefore decides from now on to strictly follow the path of the steepest descent. Whenever he sees an opportunity to follow an even steeper downward path, he adjusts his route. In this respect, the wanderer even has something ahead of the gradient descent algorithm, as will be explained later. For a differentiable function, the direction of the steepest descent corresponds to its gradient. The following algorithm describes the stepwise determination of the optimal direction and consecutive adaptation of the search route for a differentiable function f.

Algorithm (Gradient Descent)

1: **Init** $\theta_0 = \theta \in \text{dom}(f)$
2: **for** (stopping criterion is not fulfilled)
3. $\nabla\ \theta_i = -\nabla f(\theta_i)$
4. Find optimal step size ϵ_i
5: Update $\theta_{i+1} = \theta_i + \epsilon_i \cdot \nabla\theta_i$.
6: **end for**

When applying **batch gradient descent** optimization, the gradient is computed with respect to the complete training data. This is costly and supports overfitting. It is therefor useful to divide the whole training set (one **epoch**) into several smaller batches, so-called **mini-batches**. In general, the use of the terms batch and mini-batch is not consistent in literature.

Stochastic Gradient Descent (SGD)

Batch gradient descent optimization suffers from different peculiarities. As mentioned above, one of them is overfitting as it converges to exactly zero at a minimum. This minimum is the perfect fit for the training data but not necessarily an overall good fit. Stochastic Gradient

Descent however introduces noise into the learning process by sampling the observations used to compute the gradient randomly and consequently aggravates finding the *perfect* minimum.

Algorithm (Stochastic Gradient Descent)

Require: Learning rate schedule ϵ_1, ϵ_2,...
Require: Initial parameter θ
Init $k = 1$
while stopping criterion not met **do**
    Sample a mini-batch of m examples from the training set $\{x^{(1)}, ..., x^{(m)}\}$ with corresponding targets $y^{(i)}$.
    Compute gradient estimate: $\hat{g} = \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
    Apply update: $\theta = \theta - \epsilon_k \cdot \hat{g}$
    $k = k + 1$
end while

Recalling the example of the wanderer, the learning rate corresponds to the length of the track followed before adjusting the direction. While the wanderer will do so using his consciousness, it is more difficult to incorporate the adaptation of the step size into a gradient descent algorithm. A learning rate schedule often incorporates the following procedure: For $k \in \{1, ..., \tau\}$ "it is common to decay the learning rate linearly" [(7), p.291] according to

$$\epsilon_k = (1 - \alpha)\epsilon_0 + \alpha\epsilon_r = (1 - \frac{k}{\tau}) \cdot \epsilon_0 + \frac{k}{\tau} \cdot \epsilon_r. \tag{31}$$

After iteration τ, a fixed learning rate ϵ is used.
A suitable learning rate is best elected by the comparison of several learning curves. A learning curve shows the values of the objective function, more precisely, the empirical risk over time. However, behind the selection of the best learning rate there is a whole science of its own. Of course it is advantageous to adapt the learning rate with respect to the iterations instead of using a fixed schedule. The technical part of this thesis uses **ADAM** as "adaptive learning rate optimization algorithm" [(7), p.305]. In order to understand ADAM, its components are explained in advance.

Momentum

Momentum is a term used in the area of mechanical physics. Imagine yourself sliding down a slide in a water park. Typically, the slide speed will increase on the steeper sections and therefore they will be passed faster. There are functions of high curvature that would highly profit from a velocity increase and whose optimization would get stuck less likely in a local minimum. Furthermore, a speed adjustment would be advantageous in case of similar gradients with small values or if the gradients are noisy. The effect of momentum on gradient descent optimization of a concave function is displayed in the figure below.

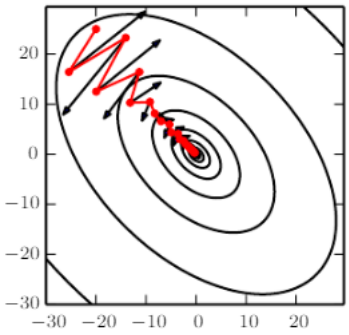


Figure 9: Illustration of Momentum (from [(7), ch. 8, p. 293])

The altitude lines indicate the steepness. They are the closer the steeper the region between. rather flat regions are passed faster with a larger step size and a smaller step size is used at the steeper sections.
The velocity v is added to the iterative algorithm (usually SGD) as follows

$$v \leftarrow \alpha v - \epsilon \nabla_\theta \left(\frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta), y^{(i)}) \right), \tag{32}$$

$$\theta \leftarrow \theta + v \tag{33}$$

The hyperparameter $\alpha \in [0, 1)$ serves to control how fast the impact of earlier gradients on the update decreases. With the ratio of α to ϵ increasing, the more past gradients influence the update of the direction. The more consecutive gradients point towards the same direction the higher is the learning rate, as postulated above. The second parameter ϵ is necessary to force convergence.
There has also developed a different kind of momentum, the **Nesterov momentum** described below.

Nesterov momentum

The development of Nesterov momentum succeeded Nesterov's accelerated gradient method. It differs from momentum in that the gradient is computed only after adding the current velocity to the parameters. The corresponding update rules look as follows:

$$v \leftarrow \alpha v - \epsilon \nabla_\theta \left(\frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta + \alpha v), y^{(i)}) \right), \tag{34}$$

$$\theta \leftarrow \theta + v \tag{35}$$

While Nesterov momentum effects batch GD in a positive way it does not seem to be advantageous in SGD optimization.

ADAM owes its name to "adaptive moments" and shows similarities to a combination of momentum with an adaptive learning rate algorithm called **RMSProp**. It was introduced to counteract the negative characteristics of a different adaptive learning rate algorithm called

AdaGrad, namely decaying the learning rate too fast. RMSProp makes use of an additional hyperparameter ρ to administer the extent of an exponentially decreasing moving average. Still there are differences towards comprehending ADAM simply as the combination of RMSProp with momentum. On the one hand, bias corrections are part of ADAM and on the other hand momentum is involved in a different manner.

ADAM

Finally, the ADAM algorithm is presented below.

Algorithm (ADAM)

Require Step size ϵ (Suggested: 0.001)
Require Exponential decay rates for moment estimates, ρ_1 and ρ_2 in $[0, 1)$
(Suggested: 0.9, 0.999)
Require Small constant δ used for numerical stabilization (Suggested: 10^{-8})
Require Initial parameters θ
    Init 1st and 2nd moment variables: $s = 0$, $r = 0$
    Init time step $t = 0$
    while stopping criterion not met **do**
        Sample a minibatch of m examples from the training set $\{x^{(1)}, ..., x^{(m)}\}$ with corresponding targets $y^{(i)}$.
        Compute gradient: $g = \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)}$
        $t = t + 1$
        Update biased first moment estimate: $s = \rho_1 s + (1 - \rho_1) g$
        Update biased second moment estimate: $r = \rho_2 r + (1 - \rho_2) g \cdot g$
        Correct bias in first moment: $\hat{s} = \frac{s}{1 - \rho_1^t}$
        Correct bias in second moment: $\hat{r} = \frac{r}{1 - \rho_2^t}$
        Compute update: $\nabla\theta = -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}} + \delta}$ (element-wise)
        Apply update: $\theta = \theta + \nabla\theta$
end while

So far, there are no recommendations regarding the best learning algorithm. However, algorithms with adaptive learning rates tend to produce rather robust results so far.

2.2.3 Error Backpropagation

This section is strongly oriented in [(7), ch.6] and [(8), ch.2].
The goal of optimization is to find the weights minimizing the error rate and hence the loss function's value. As stated earlier, neural networks correspond to a chain of functions and a function's gradient points towards the direction of the steepest descent and likely towards at least a local minimum. In backpropagation both are combined. First, it is important to note that the class of networks presented so far always works in that way that input is fed from one side and then put through all layers until the output layer. Up to this section a network was always supplied with input from one side and then the input was processed through all layers

up to the output layer. This type of network is called **Feed Forward Network** and owes its name precisely to this process. In the optimization process, however, it is advisable to first consider the deviation from the target value, i.e. to start from the back. Backpropagation aims to determine the deviation from the target value due to a parameter backwards through the network and therefore backwards through a chained function. The XOR example is used again for illustration purposes.

The optimal weight for the connection $w_{11}^{(2)}$ between $z_1^{(1)}$ and the output unit o shall be determined. The network also corresponds to the more compact notation

$$f(x) = f_{out}(z^{(1)}) = f_{out}\left(f_{in} \cdot \begin{pmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \end{pmatrix} + w_0^{(2)}\right). \tag{36}$$

Let the loss function be **Mean Squared Error (MSE)**

$$\text{MSE}(f) = \frac{1}{|X|} \sum_{x \in \mathcal{X}} |y - f(x)|^2. \tag{37}$$

The optimal $w_{11}^{(2)}$ minimizes the MSE for all $x \in \mathcal{X}$. But not all nodes in the network depend on $w_{11}^{(2)}$. Neither nodes nor connections set before the hidden layer are influenced by the choice of $w^{(2)}$ as it is applied only later. Consequently, the gradient of the MSE in regard of $w_{11}^{(2)}$ looks as follows:

$$\frac{\partial MSE(f)}{\partial w_{11}^{(2)}} = \frac{\partial MSE(f)}{\partial f_{\text{out}}} \cdot \frac{\partial f_{\text{out}}}{\partial f_{\text{in}}} \cdot \left(\frac{\partial f_{\text{in}}}{\partial w_{11}^{(2)}} + \frac{\partial f_{\text{in}}}{\partial w_{21}^{(2)}}\right) = \frac{\partial MSE(f)}{\partial f_{\text{out}}} \cdot \frac{\partial f_{\text{out}}}{\partial f_{\text{in}}} \cdot \frac{\partial f_{\text{in}}}{\partial w_{11}^{(2)}} \tag{38}$$

The same procedure works for all weights to calculate their respective gradients - as long as there exist derivatives for the used activation functions and loss. Before training, the weights are initialized randomly. However, gradient-based optimization is useful only if it is stable. Exploding and vanishing gradients both create new challenges as countermeasures are needed. In a later section, vanishing gradients will be linked to so-called long-term-dependencies expressed as rather deep functions consisting of a long chain of functions. A different model structure will be proposed to counteract the cause. Now, a measure to react in case of exploding gradients is described.

2.2.4 Gradient Clipping

The descriptions below are taken from [(7), ch.8].
Exploding gradients can occur due to the chain-wise application of functions. This gets obvious, when thinking of multiplying a number a bit larger than 1 p-times. Let for instance $p = 100$, then $(1.1)^{100} = 13780.61$ which is indeed large. A countermeasure is to clip the gradient with a norm as follows

$$\text{if } ||g|| > t, \text{ then } g \leftarrow \frac{gt}{||g||}.$$

The effect of this measure is that the step size is bounded. Large step sizes carry the risk to make the algorithm jump over the global minimum and this is where gradient clipping interferes.

2.2.5 Regularization

Because of their often large capacity due to a large number of layers or parameter expensive operations, neural networks are prone to overfitting, especially if the training data is of limited size. A regularized loss function or the addition of noise as common in the area of machine learning may provide a solution. However the network structure also includes the possibility of a further measure.

Dropout

The following explanation is taken from [(26)].
If overfitting has occurred, the cause is in general two-fold. On the one hand, the model has adapted to the training data too exactly and on the other hand, the type of adaptation was already determined by the structure of the model. For this reason it would often be helpful to have not only the prognosis of one model but the average prognosis of many models available. Since it would be very costly to fit many models, the method of dropout was developed as a less expensive alternative. Dropout requires the so-called **dropout rate** as a hyperparameter. This rate is in the interval $[0, 1]$ and indicates how many percent of the neurons in the network are randomly neglected at each update. Ignoring some neurons results in a new network in each iteration and therefore simulates the effect of averaging many models.

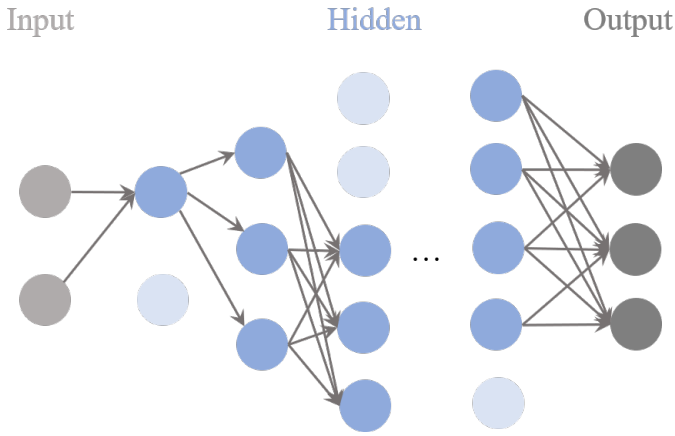


Figure 10: Visualizing the Effect of Dropout, Dropout Rate = $\frac{1}{3}$ (Own Illustration)

2.3 Word Embeddings

If numerical operations shall be applied to text, representing the text in a numerical way is consequently necessary. Popular techniques to do so and even find a continuous representation are e.g. developed by Google (Word2Vec), Facebook (FastText) and Stanford (GloVe). Word embeddings manage to reflect syntactic and semantic information together, more precisely, find words with similar meanings or a close context theirselves in proximity. And there is an additional advantage coming along the continuous representations. In comparison to *one-hot-coding* for instance whose dimensionality corresponds to the number of words inside the dictionary, a lot less dimensions are needed. As the application part of this master thesis uses a Word2Vec embedding, only Word2Vec is explained here.

2.3.1 Word2Vec

The following descriptions are taken from [(12)] and [(15)].
The proposed Word2Vec architecture was the result of Tomas Mikolov and his team members' assessment of several models regarding computational complexity and quality. It corresponds to a group of two models both listed under the name Word2Vec.
The main focus lied upon training the representations in a way such that not only would semantic closeness be reflected by spatial proximity but also should the vector space be based on "the linear regularities between words" [(12), p.2]. A linear regularity of this kind is for instance the formulation of a query as a vector operation and finding the corresponding result close to the correct answer. More precisely should for example the sum of the vector representations of "Russia" and "'River" be close to the vector representation of "Volga River" [(12), p.2].
But before the quality can be optimized, training text must be fed into the model. Models processing text data usually take text translated into **tokens** as input. This works similar for all Natural Language Processing (NLP) tasks. The tokenization takes place by first identifying unique words and counting their occurrences, then deciding on whether punctuation marks shall be dismissed and allocating numbers to words afterwards, often in accordance to their count (the higher the frequency, the lower the number). Taking for instance the following text fraction

"Eating porridge in the morning is a healthy way to start your day. Continuing with a short workout will then keep the doctor away.",

the text is first split into single words and all punctuation marks are removed. Then, replacing all words with their corresponding token will result in something like

3|4|5|1|6|7|2|8|9|10|11|12|13|14|15|2|16|17|18|19|20|1|21|22.

Earlier approaches to finding continuous word vector representations consisted of a feed forward network whose first layer would be a dense layer without activation function and therefore the network would perform a simple linear projection. The consequent hidden layer would create a non-linear feature map of the projected input. The disadvantage in training a so-called **neural network language model (NNLM)** taking tokenized text as input and simultaneously learning the word vectors as well as a distribution of the words is its high computational expense. Most of the computational effort is due to the non-linearity inside the hidden layer. This layer is discarded in Word2Vec. Therefore, a language model is not built using an encompassing single approach from the onset as a whole but a model with a modest number of parameters is used first to train the word vectors and those are fed into the NNLM only thereafter. More precisely, the process of training word vectors is now separated from learning the word distribution. Two kinds of models are recommended in this respect and both are described below.

Continuous Bag-of-Words Model (CBOW)

The name of this model is influenced by one of the bag-of-words model's characteristics, namely the neglect of word order. Continuous vectors furthermore describe a word in its semantic environment and consequently the representation is not simply bag-of-words. The weights of the projection layer are shared for all tokens as there is no further hidden layer which results in averaging the inputs. The prediction task is constructed as a log-linear classifier outputting the

word inbetween a given input of several preceding and subsequent words. The network structure for the prediction task is visualized in the figure below.

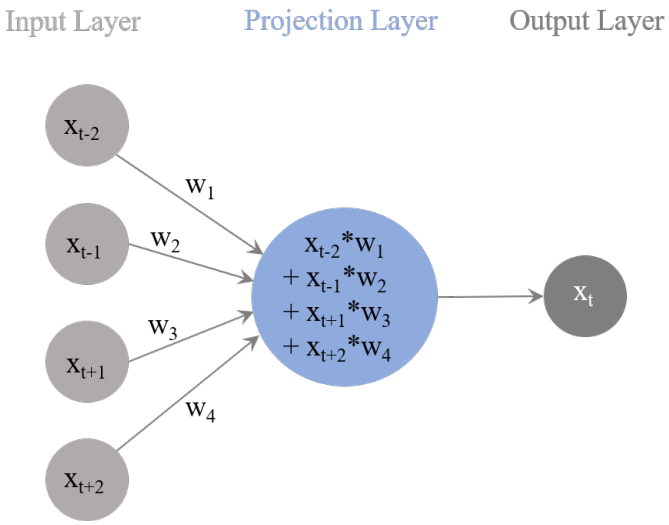


Figure 11: CBOW (Based on (15), p.3])

Continuous Skip-gram Model

The continuous Skip-gram Model owes its name to the N-Gram Model which includes the last $N-1$ preceding words to estimate the most likely successor [(31), p.4]. In contrast, Skip-Gram omits some of those $N-1$ words during training. Again, a model with one projection and no further hidden layer is trained as a log-linear classifier but now it learns to predict a specific number of context words for a given input word. The range size R indicates how many precedent and subsequent words form the target set. The larger R the better word vectors are trained but to the drawback of growing computational expenses. One way out comes with the observation that words have usually less connection to words in a further distance. Consequently, sampling R words inside a larger range C as well before as after the training word and allocating more weight to the closer neighborhood does the trick. The Skip-gram Model is visualized as a feed-forward network below.

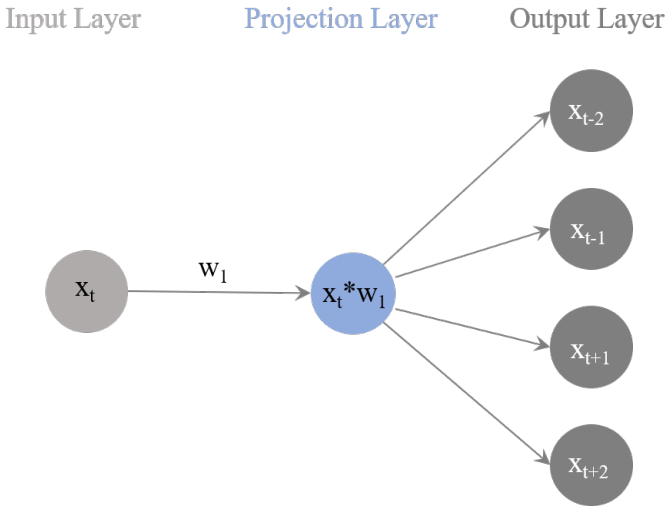


Figure 12: Skip-Gram Architecture (Based on [(15)], p.3)

In order to determine good word representations that facilitate the output of words in the context of an input word, a suitable loss function is needed. The optimization of the following negative log-likelihood

$$-\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{39}$$

leads to these representations where the high probability of finding a word in another word's context is reflected by their spatial proximity. The probability $p(w_{t+j}|w_t)$ is calculated with the softmax function which however is very costly. Therefore the Skip-Gram-Model as such does not help to reduce computational expenses. In [(12)], three methods are presented with the aim of reducing such expenses, the Hierarchical Softmax, Negative Sampling and Subsampling of Frequent Words. As their explanation would go beyond the thematic objective of this master thesis they are left out here.

2.3.2 (Paper) dpUGC: Learn Differentially Private Representation for User Generated Contents

Despite Word2Vec's advantages in terms of efficiency, finding word embeddings for large data sources still tends to be computationally very expensive. Being able to share a once developed word embedding is yet of great interest. But one has to keep in mind that the embedding's ability to reflect contextual information can become problematic if the data source contains sensitive information. For instance, in the case that Google requests are used as training data the vector representations of user names might be found close to the name of their hometown and consequently private information is exposed. *Xuan-Son Vu, Son N. Tran* and *Lili Jiang* addressed the problem of finding a representation preventing re-identification while retaining as much of the strength of interpretability as possible. Their approach [(13)] is now summarized shortly to give an impression of how differential privacy may influence embedding training.

The proposed mechanism is applied to User Generated Content (UGC) data. The dataset used contains a further incentive to research the topic of text anonymization as the access is restricted since the beginning of 2018 and an anonymization approach may support a republication.

Differential private training is applied to a Word2Vec embedding trained with the aid of the aforementioned Negative Sampling. However, the decision on when to apply differential privacy should be well thought out. Introducing noise after the determination of the embedding involves the risk of declining usability. Hence it is proposed to handle "the noise as a constraint" and "insert noise during the learning process" to "optimize both the performance of the model and its privacy" [(13), p.6]. Now, the idea is to introduce noise into the optimization by provoking a slight shift of the gradient or subgradient used. This is the approach of **DP-SGD** (Differentially Private Stochastic Gradient Descent). First, standard SGD is applied, then the 2-norm of the gradient is clipped and noise is added thereafter. Incorporating this clipped gradient $\tilde{\nabla}(f(x_i))$, the privacy mechanism is a modification of the Gaussian mechanism as

$$\mathcal{M}(D) = \sum_{i\in B}\tilde{\nabla}(f(x_i)) + \mathcal{N}(0, C^2\sigma^2 I), \tag{40}$$

with $C > 0$ denoting the clipping constant. Now, iterations of DP-SGD are conducted to find the optimal parameters. Within each iteration a so-called **privacy accountant** makes up the sum of previously and present privacy budget used to check whether given privacy conditions are still met.

The used mechanism was first proposed in [(10)]. The proof that DP-SGD does indeed fulfill the requirements of differential privacy is not trivial and can be read in the corresponding paper. While gradient clipping is actually a measure against exploding gradients, reading the paper makes it clear that clipping the gradient norm in this case serves to prove differential privacy.

In both possible cases, when the privacy budget is summed up for all training epochs as well as when the privacy budget is treated as a hyperparameter and set before training, there is a trade-off between the number of training epochs and the size of the privacy budget as the budget used grows with the number of iterations conducted. Consequently it would be useful to find ways to exploit small privacy budgets and to replace the globally set budget with a personalized one. Therefore, a personalized DP-Embedding with a suitable privacy accountant looking after the respective privacy budgets is trained as well.

The results mostly meet the expectations. There are embeddings with an acceptable trade-off between usability and privacy but the loss in utility reflects the early stopping of training due to the privacy budget increase in each iteration. Furthermore, exploration at character level provides the indication that differential privacy may sometimes work as regularization by noise addition and even improve the training process.

At this point, a possible conclusion is that despite valuable results the training still suffered from the trade-off between privacy and utility and hence it could be beneficial to try a different strategy. As privacy and data utility correspond to two counterparts, a network that not only builds on the contradiction but also exploits it to its advantage could be of use. The next section deals with networks of this kind.

2.4 Generative Adversarial Networks (GANs)

2.4.1 Introduction to GANs

The following section is based on [(36)] and [(19)].
In 1928, the scientist John von Neumann achieved a major breakthrough. He proved the so-called minimax theorem of game theory. This theorem contains a game strategy for so-called two-person zero-sum games, referring to games of two players where the winner takes the whole stake while the opponent gets nothing. Furthermore in two-player zero-sum-games, a situation that is advantageous for both players since any deviation means a worsening for both is called **Nash equilibrium**. Interestingly, the minimax strategy where each player is *maximizing* their profit under the constraint of simultaneously *minimizing* their risk turns out as Nash Equilibrium. While both terms are very common in the area of game theory, they might currently seem misplaced in the area of deep learning. But this in fact changed in 2014, following the introduction of Generative Adversarial Networks (GANs). In the following, their concept is described

based on [(24)] and [(23)].

A common difficulty in dealing with the feed forward networks presented so far is the need for labelled data. For each input there should be an associated target. However, in many situations this is not possible either because of too large amounts of data or, for example, in the area of anomaly detection, because there is no certainty at all as to when an anomaly is actually present. For this reason, it is especially desirable to be able to model p_{data} using a network. In contrast to classical classification or regression problems, the output of which is an approximation of the label, a GAN outputs elements drawn from the approximated p_{data}. It is only able to do so using a competitive training process including two models. The following is a description of how this works.

A GAN is mainly constructed of a so-called *generator* network $\mathcal{G}$ and a *discriminator* network $\mathcal{D}$. These networks are challenging each other to achieve contradictionary goals simultaneously. A typical example is to "think of one network as an art forger and the other as an expert" [(24), p.1]. When it comes to visual data for example, $\mathcal{G}$ may try to generate images as similar to the training data as possible, while $\mathcal{D}$ 'receives both forgeries and real (authentic) images, and aims to tell them apart" [(24), p.1]. The accuracy of $\mathcal{D}$ is trained with the help of an error signal indicating whether an image was classified correctly or not. However $\mathcal{G}$ learns through the competition only - it simply aims to create samples leading to a worse accuracy of $\mathcal{D}$.

Mathematically, the generator can be seen as a mapping $\mathcal{G} : z \mapsto \mathcal{R}^{\dim(x)}$ "from some representation space (latent space) to the space of the data" [(24), p.1] ($\dim(\cdot)$ indicates the respective dimensionality). For illustration purposes, assume that a data set of images showing planes is to be used to generate further images with planes. A noise-generating distribution often operates as latent space. $\mathcal{G}$ will then try to transform the given noise in a way that the outcome resembles an image of a plane. In contrast, $\mathcal{D} : x \mapsto (0, 1)$ and thus determines a probability that a given input x either belongs to the training data (value close to 1) or has been generated by $\mathcal{G}$. If the generator has learned the ability to create samples which are indistinguishable from the training data, the discriminator will achieve the same results as in random guessing and generically output a probability of 0.5. The structure is summarized in the figure below.

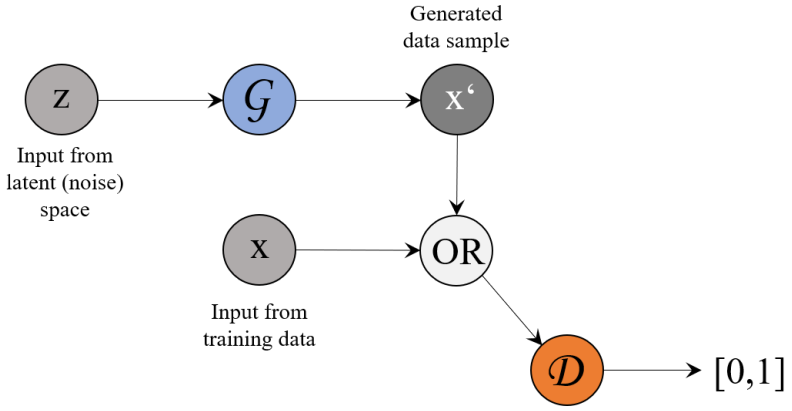


Figure 13: Basic Structure of a GAN (inspired by [(24), p.2]

Both networks can be neural networks of any kind. Feed Forward Networks are common as well as so-called Convolutional Neural Networks (CNNs) which are very popular when it comes to visual data.

Training a GAN

To determine optimal parameters for both models the contradictory goals must be set into a joint loss function. This is achieved by minimizing a loss function under the data distribution induced by $\mathcal{D}$ and maximizing it under the data distribution induced by $\mathcal{G}$:

$$\max_{\mathcal{G}} \min_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}). \tag{41}$$

More precisely, these contradictory objectives can be expressed as

$$V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \log \mathcal{D}(x) + \mathbb{E}_{z \sim p_z(z)} \log(1 - \mathcal{D}(\mathcal{G}(z))). \tag{42}$$

The minimization of this joint loss function then reflects both contradictory goals. On the one hand, $\mathbb{E}_{z \sim p_z(z)} \log(1 - \mathcal{D}(\mathcal{G}(z)))$ is minimized which corresponds to guiding the generator model towards developing samples of such a good quality that $\mathcal{D}$ outputs a high probability for them to belong to the real data. On the other hand $\mathbb{E}_{p_{\text{data}}(x)} \log \mathcal{D}(x)$ is maximized so that the discriminator is forced to identify samples of the real data distribution with high probability. Goodfellow et al. have proven that

$$\mathcal{D}^*(x) = \frac{p_{\text{data}(x)}}{p_{\text{data}(x)} + p_g(x)} \tag{43}$$

is optimal for a given generator while every optimal generator model fulfills $p_g(x) = p_{\text{data}(x)}$. When assuming the perfect generator, $p_g(x)$ can be replaced by $p_{\text{data}}(x)$ in the above formulation of $\mathcal{D}^*(x)$. The outcome indicates that the discriminator is no longer able to make better predictions than $p = 0.5$ for all inputs and is consequently as confused as possible. This point of the optimization process corresponds to the Nash Equilibrium when considering the adversarial training as a two-player game between $\mathcal{G}$ and $\mathcal{D}$.
The optimization is conducted separately or, more precisely the parameters of one model are updated while the parameters of the other one are held fixed. It would be best to let the discriminator after each iteration of the generator train until it has learned again to distinguish perfectly between real and fake. However, it is common practice to adjust the discriminator only a few iterations because otherwise the training would be too difficult for the generator. The number k of $\mathcal{D}$-iterations after one $\mathcal{G}$-iteration is in fact a hyperparameter. In general, the training of a GAN is known to be unstable and complex and it is often difficult to converge to equilibrium. But as regards each problem setting there are usually specific measurements to enable better training.

2.4.2 (Paper) Learning Anonymized Representations with Adversarial Neural Networks

The inspiration to translate the contradiction of privacy and data utility into an adversarial loss as proposed in the application part in a later chapter, comes, amongst others, from [(27)]. The according paper is summarized shortly below.

The main idea of the paper is the construction of an adversarial network in order to encode sensitive inputs. As in all previous settings, the encoding shall be performed in a way that all relevant characteristics apart from the private ones are kept. The GAN will play the two conflicting goals of privacy and data utility off against each other.

For a dataset X, a representation U, „regular labels“ Y and „private labels“ Z the provided network structure looks as follows:

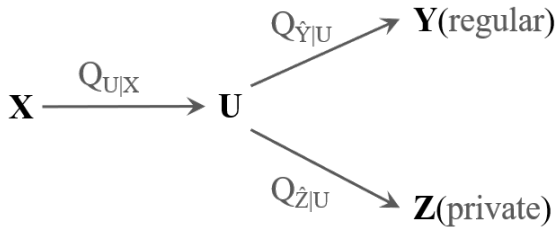


Figure 14: Construction of Adversarial Framework (own illustration)

The first classifier is used to assess the degree of privacy by measuring the amount of mutual information between input and output while the second evaluates how well *regular* attributes are preserved. $Q_{\hat{Y}|U}$ is therefore optimized with regard to the cross-entropy-loss. The adversarial loss incorporates the losses of both prediction branches $Q_{\hat{Z}|U}$ and $Q_{\hat{Y}|U}$. It aims to force the encoder to focus on those attributes identifying a person as it decreases only if the remaining data utility is high. Therefore, this approach is different from the previous ones in that it is not differential privacy that is applied to the data and in that the remaining utility is therefore evaluated only afterwards. Instead, the trade-off participates already actively in the training process.

The Loss Function

The paper contains the complete derivation from an expression of the trade-off between the two misclassification probabilities of the private and regular label classifier to a combined loss function. Only the main characteristics are described here.

The trade-off between privacy and data utility may be reformulated as requiring the misclassification probability of $Q_{\hat{Y}|U}$ to be low while the misclassification probability of $Q_{\hat{Z}|U}$ shall be greater than at least $1 - \epsilon$. For instance, $\epsilon = 0.5$ corresponds to random guessing if there are two persons to classify. Exploiting an upper respectively a lower bound for the misclassification

probabilities, the loss function can be expressed as

$$\min_{(Q_{U|X}, Q_{\hat{Y}|U}) \in \mathcal{F}} \left\{ \mathcal{R}_{\text{emp}}(Q_{\hat{Y}|U}, Q_{U|X}) + \lambda \cdot \mathcal{I}(\hat{P}_Z; \hat{Q}_{U|Z}) \right\}, \tag{44}$$

where $\mathcal{I}(\hat{P}_Z; \hat{Q}_{U|Z})$ denotes the mutual information of the true private label distribution and the distribution induced by $\hat{Q}_{U|Z}$. Mutual information is a common term in information theory and its mathematical definition simply expresses the obvious meaning. λ takes on the role of a control parameter for the privacy-utility-trade-off. From the formula it gets obvious that the larger λ the more privacy and the less mutual information is targeted for. The loss function is still not tractable because of the mutual information relying on a joint probability. Further amendments are made but all operations are of a mathematical nature and provide no further understanding of how to include a privacy condition into the loss function. As a consequence, further derivation is discarded here.

The authors propose to train the encoder as follows:

1. The encoder $Q_{U|X}$ and the regular label branch $Q_{\hat{Y}|U}$ are optimized together with respect to the regular label cross-entropy.

2. The private label predictor $Q_{\hat{Z}|U}$ is optimized according to its cross entropy loss, the encoder is held fixed in the meanwhile.

3. The adversarial training alternates between

   - sampling a batch of N training observations and update both branches and
   - sampling a batch of N training observations and update the encoder.

In the alternation between updates of the opposing branches the adversarial training can be read from the procedure.

The overall conclusion is that the data utility still suffers the more the level of privacy increases. But in both example tasks, digit classification and sentiment analysis, an acceptable level of usability is preserved. Finally, it is again important to emphasize that the proposed method is different from differential privacy. Privacy is reduced with the mutual information depending on the dataset and therefore without a generically applicable form. Furthermore, no randomness was introduced.

2.4.3 (Paper) Generating Differentially Private Datasets using GANs

A second approach implementing the conflict of privacy and data utility into a GAN is described now. All explanations follow the corresponding paper [(22)]. In contrast to the precedent paper, a differential privacy mechanism as presented in the section about privacy is used, more precisely the **Gaussian noise mechanism**

$$\mathcal{M}(d) \triangleq f(d) + \mathcal{N}(0, s_f^2 \cdot \sigma^2). \tag{45}$$

Again, the variance of the Gaussian distribution $\mathcal{N}(\cdot, \cdot)$ is the product of a σ^2 and the L_2-sensitivity of the function f.
The goal of this paper is the development of a mechanism to enable differentially private data release, especially to third party machine learning providers. The users can apply the resulting mechanism theirselves and therefore the need to trust a third party is redundant. So far, the paper was declined as a conference paper at ICLR 2018 because several issues were detected. Nonetheless, the main idea is described shortly. Below, the structure of the GAN is illustrated.

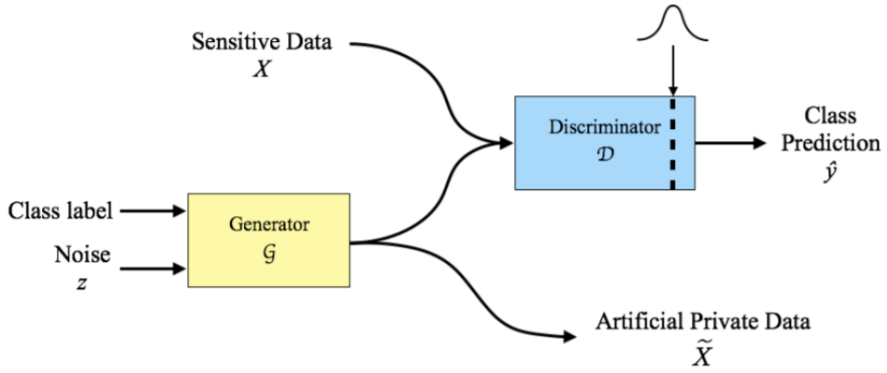


Figure 15: GAN of Proposed Structure [(22), p.4]

The dotted line in the discriminator shows the central idea of adding a Gaussian noise layer into the discriminator part. This measure is based on the hope that the output of $\mathcal{D}$ and thus the updates of $\mathcal{G}$ are differentially private then in comparison to the input data X. All corresponding proofs are listed in the paper but as mentioned their validity seems to be questionable. Irrespective thereof, the results are interesting. First, a so-called **teacher** model was trained. This teacher is the proposed GAN. Next, a **student** model is trained as a classifier on the generated data. The student model is not restricted to any model class and provides a double advantage, i.e. automated assessment of the output quality of $\mathcal{G}$ and the comparability of the test errors.
Two popular tasks were used to evaluate the remaining data utility, the MNIST dataset with images of hand-written digits and the SVHN dataset containing photographs of house numbers collected by Google Street View. The results seem indeed rather promising as an adequate level of data utility is preserved at different privacy budgets.

The application section below is also based on an adversarial loss but due to the building blocks (differential privacy and attention mechanisms) a GAN structure in the regular sense turned out not viable. However, an adversarial loss will be used and the model presented below incorporates a multi-objective-loss of this kind.

2.4.4 (Paper) Mobile Sensor Data Anonymization

Amongst others, artificial intelligence supports the fitness of people. Accelerometers and gyroscopes are examples of motion sensors helping to record motion patterns in order to quantify sporting results. As those patterns are rather unique, re-identification of a user is feasible when the sensor data is shared. In contrast, the utility to be kept corresponds to the recognition of spe-

cific activities. In [(28)], a so-called **Anonymizing Auto-Encoder (AAE)** is proposed in order to amend the sensor data accordingly. To understand the term auto-encoder, encoder-decoder networks are briefly introduced first, in accordance with [(11), p.3-4]. Then, autoencoders and the proposed AAE are represented according to [(28)].

When thinking of a translation task, a network will need to first find a representation of the provided input, for instance a sentence in English. A new representation is determined either to reduce the input's dimensionality by first identifying and then concentrating on the most important features only or to extend its informational content by practically unfolding the initial form into a higher-dimensional space. This is called **encoding** and all layers performing encoding represent the **encoder** part of the network. The output sentence however, is French. The encoding is then again transformed but now in a way that it represents the French translation. This procedure is called **decoding**.

An auto-encoder is characterized in that encoding and decoding are fitted for the same in and output. The AAE shall be placed at the user's device to encode sensitive data before sharing as presented in the image below.

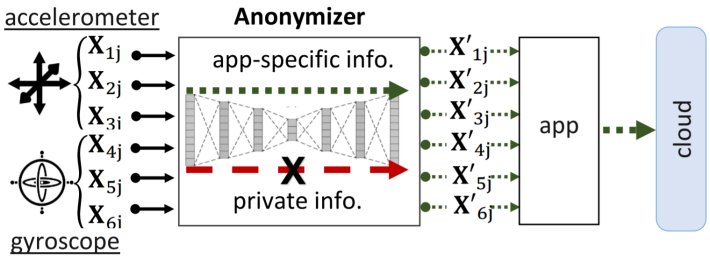


Figure 16: Anonymization before Data Sharing [(28)]

The anonymization technique is not based on differential privacy as every DP-mechanism is based on an overall data set and this would involve prior data sharing. In contrast, as in the earlier explained paper, an information-theoretic approach is chosen and the mutual information between the anonymized data and the true user names is minimized.

The Loss Function

The multi-loss function reflecting the desire for anonymization together with the preservation of the sport specific patterns under the additional condition of only minimal overall change of the data, has the following form:

$$L = \beta_i L_i + \beta_a L_a + \beta_d L_d. \tag{46}$$

All hyperparameters β target the trade-off between the three kinds of losses. The loss of user-specific information L_i is measured with respect to the mutual information as explained before. L_a corresponds to the categorical cross-entropy-loss in the activity classification model. Finally, L_d measures the discrepancy between input data X and anonymized output X' as a distance.

2.5 Recurrent Network Architectures

2.5.1 Introduction to Recurrent Networks

So far in this master thesis, networks have only processed an input tensor of fixed size. But when it comes to unstructured data types as text documents it is advantageous to use a more flexible architecture that additionally pays respect to the input order. A respective network structure is presented based on [(7), ch.10] and [(39)]. Recurrent neural networks (RNNs) take an input sequence of variable-length and output a fixed-size encoding of the input. They accomplish so by reading in parts of the input sequence one at a time, each of equal size and each at a time step t. For example, a book was read in word by word which also corresponds to how a person would read. Continuing the example of reading a book word after word, at each time step, sort of a summary of all precedent words (the **context**) is combined with the current word and a joint representation is produced. Thanks to the sequential reading process, the order of the words implicitly influences this summary. At any time step, the summary representation is of fixed-length. The model owes its ability to read sequences of variable length to the fact that the same parameters θ are used for each new representation of a previous context and a current word. The parameters are said to be **shared** among all hidden layers and for each word the hidden layer is simply reproduced. The procedure is recursive as each computation involves the computation in the previous time step. The figure below aims to illustrate the network architecture.

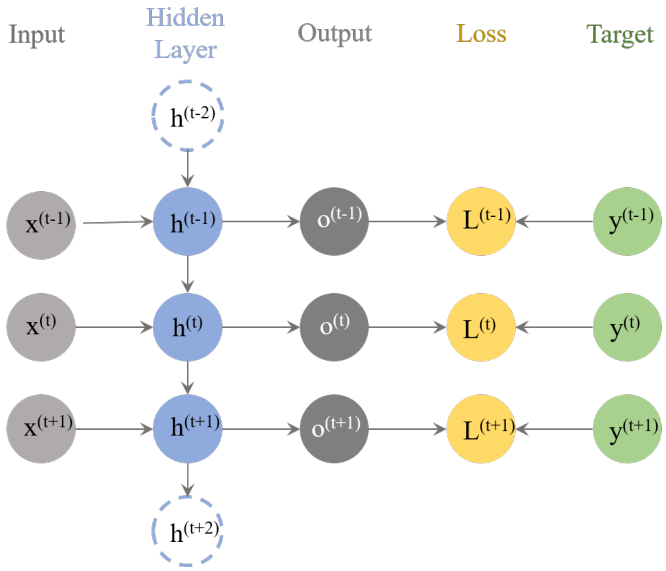


Figure 17: The RNN Architecture, inspired by [(7), p.373]

While the value of hidden unit at timestep t is computed as

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta) \tag{47}$$

actually all previous computations back to the first one ($h^{(0)}$) are implicitly contained inside the

equation:

$$\begin{aligned} h^{(t)} &= f(h^{(t-1)}, x^{(t)}; \theta) = \\ &= f(f(h^{(t-2)}, x^{(t-1)}; \theta), x^{(t)}; \theta) = \\ &= f(f(f(h^{(t-3)}, x^{(t-2)}; \theta), x^{(t-1)}; \theta), x^{(t)}; \theta) = \\ &= (...) \end{aligned} \tag{48}$$

From this formula it gets obvious that this is only possible thanks to the shared weights θ. The optimal shared weights are again approximated with backpropagation. In order to pay respect to the recursive nature of the chain function it is called **Backpropagation through time (BPTT)**. Unfortunately, regarding the optimization an obstacle becomes obvious from the shown recursivity. On the one hand, the longer a sequence the larger is the computational effort as the model is trying to *memorize* all precedent history. This results in **long-term dependencies** whose practical relevance is sometimes questionable. On the other hand complications as vanishing or exploding gradients are likely to arise. One solution for exploding gradients is the previously explained measure of gradient clipping. A version of the basic recurrent network which was deepened in order to counteract the causes of the gradient behaviour is called Long-Short-Term Memory (LSTM) and is presented further below.

2.5.2 Bidirectionality

Up to this section, there were only forward connections in a recurrent network described. In a task where the RNN reads in text, the context representation is consequently performed with regard to all words before the current input word only. This ignores however the fact that a word within a sentence is usually not only in context to the preceding words but to all words in the sentence respectively to words in preceding and subsequent sentences also. Bidirectional RNNs aim to solve this issue by including an additional hidden layer in order to read in the input in reverse order also. The consequent network structure is displayed in the figure below.

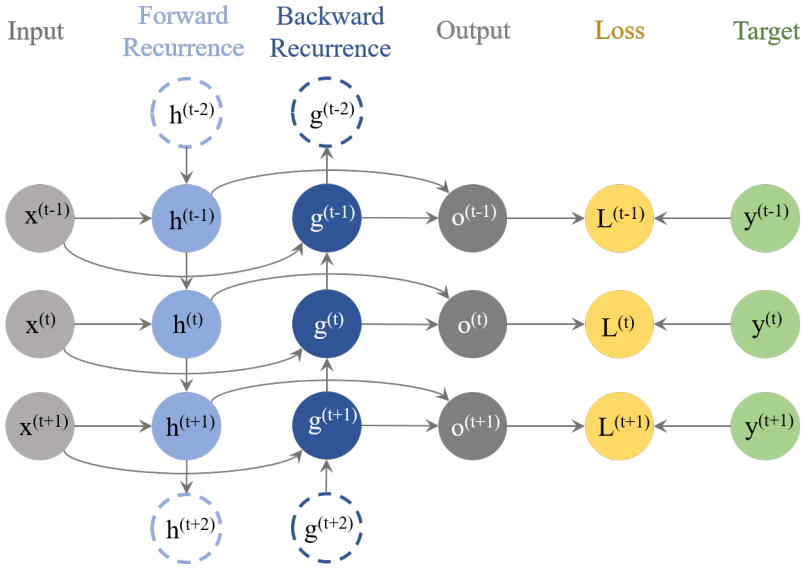


Figure 18: A Bidirectional RNN, inspired by [(7), p.389]

There are now two seperate hidden layers at each time step whose outputs are both input to the next layer (in the above figure the next layer corresponds to the output layer).

2.5.3 Long-Short-Term Memory (LSTM)

As mentioned before, recurrent networks are likely to suffer from vanishing or exploding gradients. The reason for the impractical gradient behaviour lies mainly in long-term dependencies arising when reading in long sequences. When recalling the example of reading a book, it is safe to say that in order to understand a sentence, not all words are equally important. For example, if you want to use a recipe to decide whether a dish is sweet or salty, you will actually need a fraction of the text only. Ingredients affect the taste but all other words have no influence on the result. It would therefore be of use to be able to **forget** some of the text. **Long-Short-Term Memory (LSTM)** Networks are examples of **gated RNNs** that incorporate a mechanism to discard information. There is a reduced variant of the LSTM also, called **Gated Recurrent Unit (GRU)** but only the LSTM is described here. In addition to the sources listed above, the section is in addition based on [(38)].

In LSTMs, the main idea is to optimize a network with respect to the best paths through time and therefore the best rule for forgetting. Mechanisms of such kind are implemented as self-loops inside the hidden layer at each time step. The self-loop has no fixed form but depends on the context. The mechanism is illustrated in the figure below.

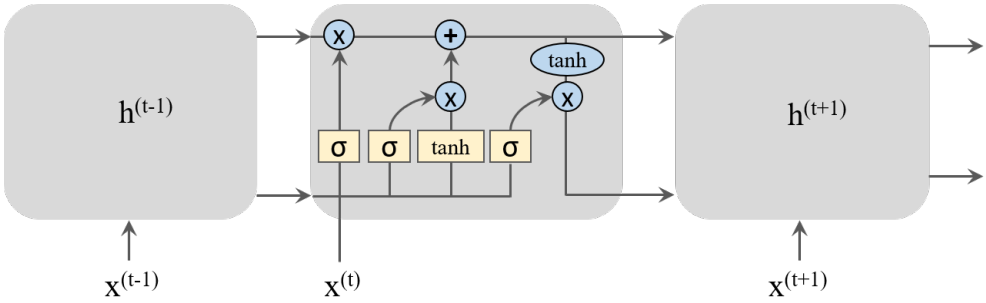


Figure 19: Exemplary LSTM Architecture, inspired by [(38)]

As displayed in the figure, the LSTM differs from an ordinary RNN in that an inner recurrence happens inside each LSTM cell in addition to the outer recurrence. Hence, more parameters are added to those of the ordinary RNN. Inside the LSTM cell, there are several **gates** with different tasks. First, the **forget gate** $f_i^{(t)}$ takes weighted projections of the current input x_t and the precedent hidden state $h^{(t-1)}$ and applies a sigmoid σ to their sum:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right). \tag{49}$$

b^f, U^f and W^f refer to the forget gate specific bias, input weight and recurrent weight matrices. Next, the **external input gate** unit $g_i^{(t)}$ is computed as

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right). \tag{50}$$

The computation is highly similar to the forget gate computation but differs in the gate specific bias and weights b^g, U^g and W^g. Now, the **internal state** $s_i^{(t)}$ is updated involving $f_i^{(t)}$ and

$g_i^{(t)}$:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma \left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)} \right) \tag{51}$$

Last, the LSTM output is controlled involving the **output gate** $q_i^{(t)}$. The value of the output gate unit is computed as

$$q_i^{(t)} = \sigma \left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^(t-1) \right) \tag{52}$$

and plays then a role in the following calculation of the hidden unit $h_i^{(t)}$:

$$h_i^{(t)} = \tanh \left(s_i^{(t)} \right) q_i^{(t)}. \tag{53}$$

An alternative method saving computational efforts when it comes to long-term dependencies in comparison to the LSTM is described within the next section.

2.6 The Concept of Attention

2.6.1 Attention in Recurrent Networks

The following section is strongly oriented in [(5)], [(11)], [(14)], [(29)] and [(35)].
Recalling how recurrent networks handle sequential data, namely by sequentially constructing a representation including the information of precedent inputs, information is sometimes kept over long connections. With the length of such connections the number of parameters needed increases accordingly and so does the length of chained functions. This results in small weights leading to gradients with small entries when it comes to optimizing those weights. The smaller the gradients, the smaller improvement is influenced by one gradient descent iteration and hence the algorithm may get stuck. This complication is called the **Vanishing Gradient Problem**. While LSTMs and GRUs try to tackle the challenges coming along with long-range dependencies, they still lose some efficiency as their models have to be able to reflect the longest dependencies and are too complex for shorter inputs as a consequence. Additionally, further computational expenses are required to calculate these partly unnecessary weights. In order to address the problem of vanishing gradients, inflexible representations and to partially avoid computational efforts, attention mechanisms were introduced by Bahdanau et al. in 2015 and published as part of [(29)].
Image Captioning represents a task that highly profits from attention mechanisms. When looking at a picture and trying to describe it, a human would not pay attention to all parts of the image at once. They would rather focus on different parts after each other and as they most likely have already explained images before, they would be following a certain recipe. Such a recipe could be to concentrate on the surrounding first, and then describe the events within. Attention tries to imitate this process by involving specific linear projections in different layers, expressing which parts of the input are especially important during a specific phase.

Fig. 20 shows exemplarily which parts of a picture an attention network attends to in order to

formulate a content description. The network will learn English syntax over time and as it must produce a sequential output it will also learn to first attend to those parts of the image referring to the first word of a sentence the most likely. The linguistic rules of the target output thus influence the order in which sections of the picture are viewed and this surely holds efficiency.

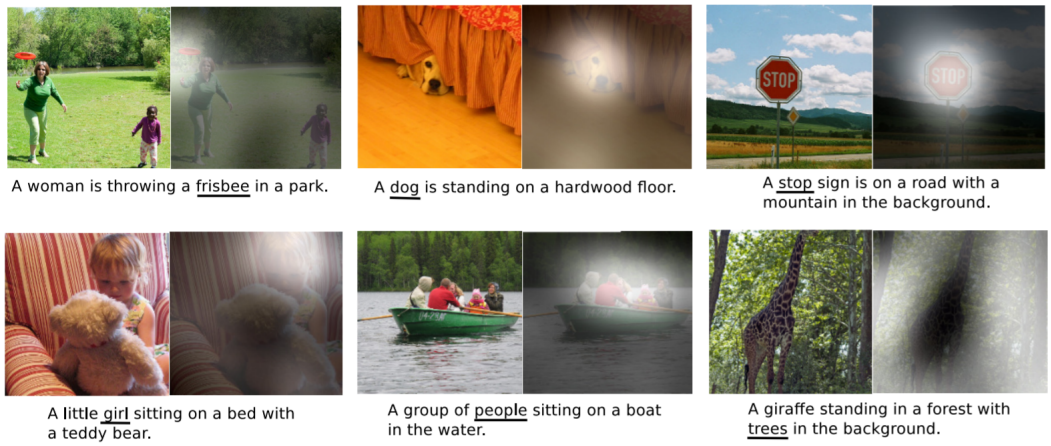


Figure 20: Image Captioning with Attention-Based Networks [(35)]

As attention mechanisms got famous in translation tasks, they were implemented into recurrent networks first. The mechanism is displayed in the figure below.

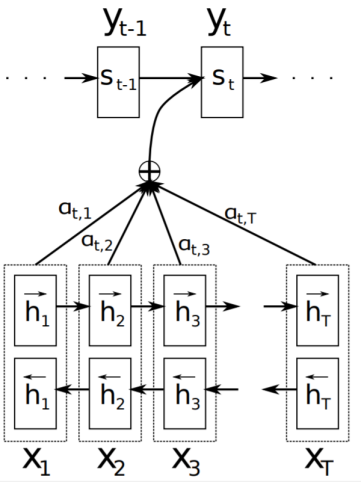


Figure 21: Attention Mechanism in a Recurrent Network [(29)]

Inside the recurrent network, attention is calculated as a weighted sum over hidden units. More precisely, a new context representation $c^{(i)}$ in order to especially support the generation of output $o^{(i)}$ is at time step i calculated as

$$c^{(i)} = \sum_{t=1}^{\tau} \alpha_{it} h^{(t)}. \tag{54}$$

Each weight α_{it} indicates the importance of hidden state h^t at output time step i. To calculate the respective weight, a so-called **alignment model** is needed first. The alignment model determines scores describing the link between a hidden unit $h^{(t)}$ and an output unit $o^{(i)}$ as follows:

$$e_{it} = a(o^{(i-1)}, h^{(t)}). \tag{55}$$

The alignment model corresponds to the later introduced **compatibility function**. Each respective weight is then computed with a softmax function:

$$\alpha_{it} = \frac{\exp\left(e_{it}\right)}{\sum_{k=1}^{\tau} \exp\left(e_{ik}\right)}. \tag{56}$$

Due to the softmax operation all entries of the attention matrix are greater than zero and sum up to 1. In addition, the weighted sum is easily differentiable.
There is furthermore the distinction between **local** and **global** attention, depending on whether all hidden states or only a local subset are regarded in the softmax. The difference becomes clear when the example of reading a book is taken up again. In order to understand the current sentence, it is usually sufficient to be aware of the context of the corresponding page and thus to grasp a local context. However, if the book is to be summarized, the whole content of the book is important and a global context representation must be created.

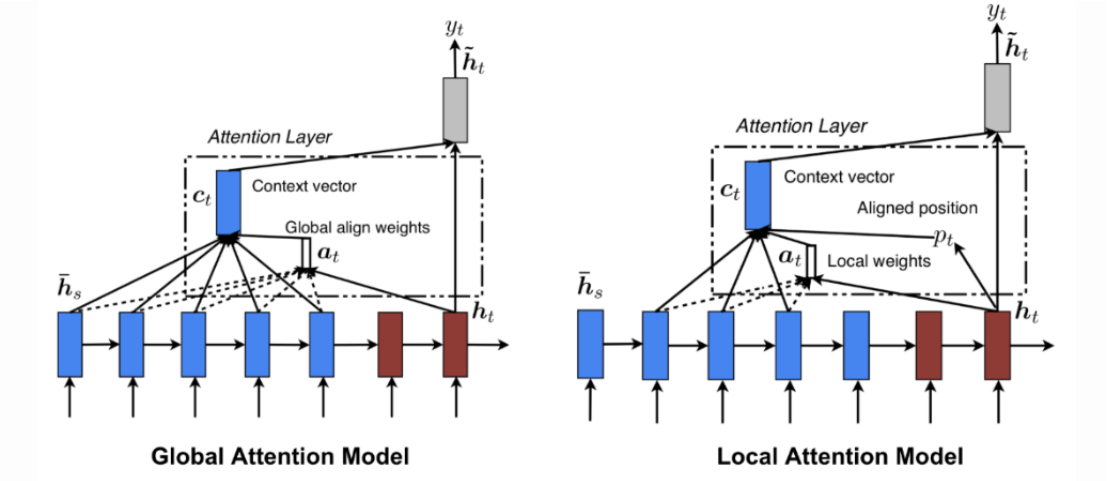


Figure 22: Global and Local Attention [(14)]

The terms **soft** and **hard** attention also refer to either weighting all hidden units and therefor only emphasizing some of them more than others (soft) or discarding all hidden units apart from a subset when determining the representation (hard).
However, in addition to the desire to avoid the computational expenses still coming along with all kinds of recurrent networks, there may be situations in which non-recurrent networks are simply better suited like convolutional neural networks in image processing. The solution to the problem of including attention in non-recurrent networks is called **self-attention** and was introduced by Google as part of the Transformer model in 2017.

2.6.2 The Transformer

All following explanations are based on the corresponding paper [(3)] unless indicated differently.

The Transformer is described as most likely "the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution" [(3), p.2].

As remarked above, it is in general tasks involving a sort of alignment, more precisely tasks which rely on structural dependencies between input and output, that profit from an efficient way to draw the respective connections. Transformers were therefor first developed to improve translation tasks and reached significant better scores than all previous models on the WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks. Additionally, the model structure offers new ways of parallelization and thus a considerable amount of computational effort can be saved. As in many translation tasks, the Transformer's structure corresponds to a **encoder-decoder-model**. The Transformer consists of a sequence of blocks of layers for both the decoder and the encoder part. The model works in a auto-regressive way as in each step the current input and the newly created representation are added together. The exact model structure is not of specific interest to this master thesis but for the sake of completeness shown in a figure below.

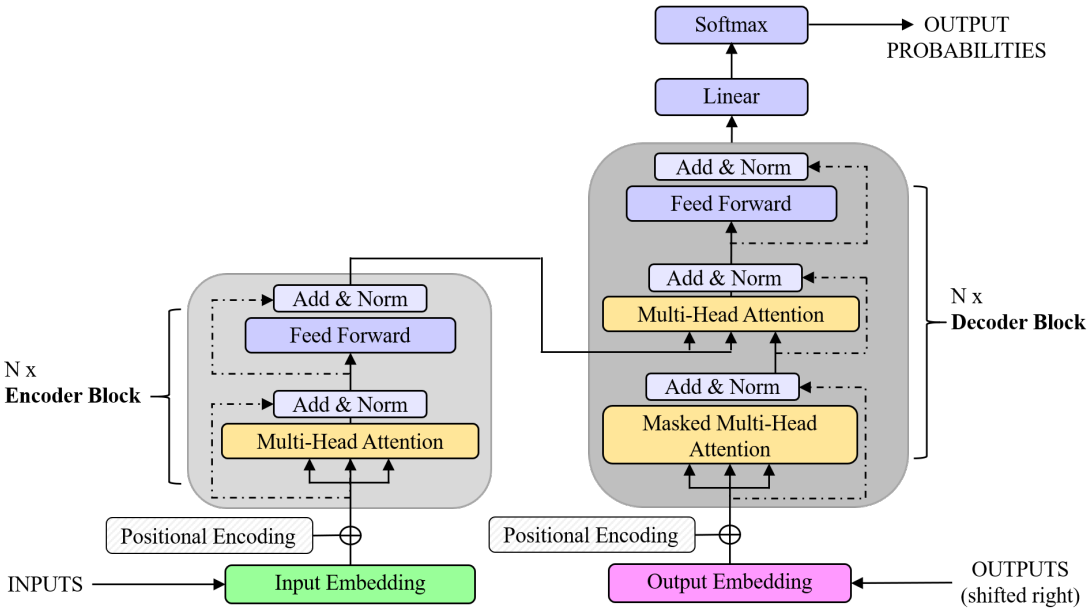


Figure 23: Transformer Model Architecture [(3)]

The original Transformer consists of $N = 6$ encoder and decoder blocks. After adding the input and output word embeddings to the model, there is an additional operation called **positional encoding**. In a recurrent network, the word order in a sentence is submitted automatically thanks to the sequential data processing. But when reading in more than one word at a time as it is the case in a non-recurrent network like the Transformer, information about their order must be stored differently. The authors propose to use trigonometric functions of different frequencies for the purpose:

$$\begin{aligned} \mathrm{PE}_{(\mathrm{pos},2i)} &= \sin(\mathrm{pos}/10000^{2i/d_{model}}) \\ \mathrm{PE}_{(\mathrm{pos},2i+1)} &= \cos(\mathrm{pos}/10000^{2i/d_{model}}) \end{aligned} \tag{57}$$

While d_{model} refers to the depth of the word embedding, i describes the dimension and pos the word's position within the input text. The resulting encoding can simply be added to the embedding as both are d_{model}-dimensional.
All prerequisites for the model are explained now and finally the integrated attention mechanism can be described.

Scaled Dot-Product Attention

Recalling the application of attention in a recurrent network, namely sequentially applying softmax to the output of the precedent hidden layer, such procedure must now be transferred to a non-recurrent model. The main purpose of attention is to draw connections - these may for instance link input and output or the input with itself. In order to find more general terms, the authors define attention functions „as mapping a query and a set of key-value pairs to an output“ (p.3). However, there are many tasks where it does not make sense to speak of key-value pairs as neither keys and values are the same nor is the query formulation strict. In self-attention it is particularly redundant, as all of them are the same and a current representation is attending to its own parts. Nevertheless, the following definition uses the query-key-value notation with Q indicating a matrix containing a batch of query vectors as columns, K the corresponding matrix of key vectors and V the corresponding matrix of value vectors. With d_q, d_k and d_v, the respective amount of vectors is denoted. Next, it is necessary to decide on a **compatibility function** to describe the relations among Q, K and V. With the dot-product as compatibility function, attention is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{58}$$

Scaling with $\frac{1}{\sqrt{d_k}}$ helps to prevent $\text{softmax}(QK^T)$ to take large values causing small respective gradients and consequently aggravating the optimization process. Below, there is a figure showing Dot-Product-Attention in the style of a graphical model for better illustration purposes.

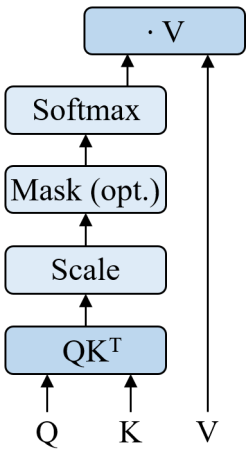


Figure 24: Dot-Product-Attention (based on [(3)])

Additive attention could replace the dot-product attention but is computationally more expensive and slower. In additive attention, the compatibility function is replaced with a sum. However, for large dimensional key vectors it works better than dot-product attention without scaling. It can be implemented as a feed forward network with one hidden layer linking the queries and key-value pairs to an output.

Multi-Head Attention

Applying the above attention function neglects available information about the keys, values and

queries as only the model's dimension is used. In order to remedy this suboptimal situation, the authors recommend linear projections of the keys, queries and values. More precisely, as described above a scaled-dot-product-attention is applied but *num_heads* times instead of one. In the tensorflow implementation of the transformer model [(37)], the embedding depth is splitted *num_heads* times and scaled-dot-product-attention is then applied to the dimensionally reduced queries, keys and vectors. Each such attention output is then called an **attention head**. In a last step, they are again concatenated and the input dimension is restored. After concatenation of all attention heads and a further projection, one arrives at the final output:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$
$$\textit{with } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

The projections are conducted by multiplication of Q, K and V with respective parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_q}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and finally the concatenated attention heads are multiplied with $W^O \in \mathbb{R}^{num\_heads \cdot d_h \times d_{\text{model}}}$, where d_h corresponds to the (reduced) dimension used ($d_h = d_v/h$) to create the attention heads.

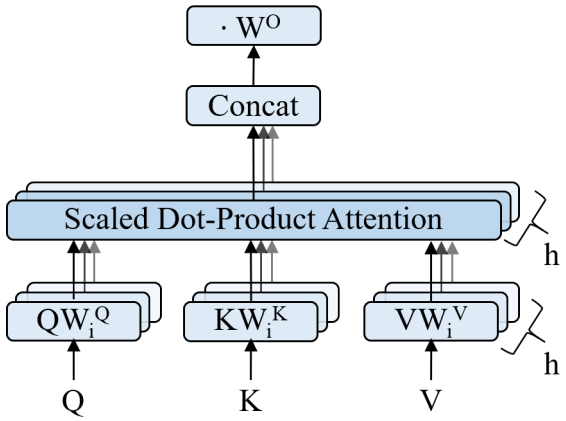


Figure 25: Multi-Head-Attention (based on [(3)])

Comparison of Computational Efforts with and w/o Self-Attention

The table presented is oriented in [(3), p.6].

Layer Type	Complexity per Layer	Seq. Operations	Max. Path Length
Self-Attention	$O(n^2 \cdot d)$	O(1)	O(1)
Recurrent	$O(n \cdot d^2)$	O(n)	O(n)
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$

According to [(37)], splitting the embedding dimension into different heads certainly holds the additional advantage of reduced computational effort.

2.6.3 BERT

End of october 2018, Google launched a pretrained open-source model in order to facilitate NLP tasks. Their follwing description is based on [(33)]. The model is called **Bidirectional Encoder Representations from Transformers (BERT)** and aims to tackle the common

problem of too low amounts of training data. Instead, BERT is pretrained on a corpus containing all Wikipedia text in sort of a revolutionary way. While common word embeddings transfer contextual similarities into space, BERT further extends the inclusion of context into the representation and is called „the first deeply bidirectional, unsupervised language representation“ model. The bidirectionality refers to including precedent and subsequent words into the representation building. Everybody can then use the pretrained BERT and conduct fine-tuning with their own task-specific smaller data sets. The combination with the pretrained model has proven to outperform training from scratch, i.e. in tasks like sentiment analysis and question-answering.

3 Development of an Attention-Based Anonymization Algorithm

Several approaches to anonymization were presented in the previous chapter. The idea to transfer the contrary goals of reaching high level of both, data utility and privacy, into an adversarial loss is central to the approach described now. Two amendments were made in comparison to the previous attempts. On the one hand, a self-attention mechanism is added in order to attend to the parts of the text containing the most sensitive data. On the other hand, the privacy descriptions as well as the algorithm applying $d_{\mathcal{X}}$-privacy to bags of words in accordance with [(21)] are utilized with the equal sized bags-of-words exchanged for documents of a standard length and additional multiplication with an attention matrix. As far as informed, this is the first approach to combine attention and differential privacy. The resulting anonymization technique is applied on the famous Enron email dataset. In the following, the data set is explained first, secondly the word embedding, then the theory that makes up the anonymization strategy and finally the models used. All networks are constructed using the deep learning library keras (version 2.2.4) with a tensorflow backend (version 2.0). The following notation will accompany the explanations:

Notation	Meaning
max_email_len	standard email length
hidden_size	size of the embedding vectors
dp_eps	privacy respective ϵ
num_heads	number of attention heads
padding_value	value used to fill emails up to standard length
num_classes	number of sendersheight

Table 1: Notation in the Application Part

3.1 Enron Email Dataset

3.1.1 The History of Enron Corporation

The famous story behind the publication of the Enron Email Corpus is described according to [(17)] and [(18)].

After the merger of two smaller energy companies in Texas, Enron Corporation was founded in 1985. From 1996 to 2001, Enron was recognized annually by Fortune Magazine as “America’s most innovative company”. The reason for this was that the company had begun to focus not only on energy but also on a well-positioned fund. Unfortunately, the fund was not as profitable as shown and over time it became apparent that billions of dollars of debt had been disguised using a broad band of accounting tricks. Enron was officially declared bankrupt on December 2nd in 2001. The particularly deep fall from the acclaimed business strategy to the company’s dissolution, made the company all the more famous. Following the declared bankruptcy, the Federal Energy Regulatory Commission conducted numerous investigations. In the course of these investigations, over 500,000 emails, mostly sent by senior management, were published. In 2015, a new version of the dataset was released to counteract the public speculation about blame. However, the relationship between sender and text has been preserved which will be used to explore anonymization in the present master thesis.

The version of the email corpus provided on kaggle as .csv file was processed. It is based on the version previously published in [(18)].

3.1.2 Amendments to the Corpus

First, the corpus had to be cleansed in order to generate some standardized format. *R Studio* was used to perform those adaptations. Several string split operations were conducted to receive a large table containing a sender name of the form "allen-p" in the first and the corresponding message text in a second column. Since some emails were forwarded, the message text was sometimes containing a sequence of emails and addresses. After several unsuccessful approaches to preserve only the first email in the sequence (if the first one was actually written by the sender), all forwarded emails were discarded. Next, several word and symbol filters were applied to erase obvious spam mails. Finally, all empty emails as well as all emails from senders outside of Enron were filtered out.
After the standardized format was obtained, it was observed that many of the remaining emails were impractically long with a length up to 11,000 words. Following a recommendation of professor Heumann, the emails were not simply shortened to a standard length, but split several times. This procedure had two weaknesses, however. Firstly, the possibility of strong correlations between the split emails arose and secondly, the original relationships between a sender and the number of emails they sent would be greatly distorted. In order to encounter these complications, all e-mails were split in a way such that only the first and the last part remained. This has the advantage that the salutation and the farewell formula, which probably support the sender identification the most, are preserved. It was experimented with the lengths 200 and 100 words. After application of further filters, 312789 emails of *max_email_len*=200 respectively 370841 emails in case of *max_email_len*=100, were available for further usage. To enable sender identification even further, one of 12 different farewell phrases containing the sender name were appended to all emails using a random mechanism. In order to not reduce the identification mechanism to only read the last word, the name is put at a different position in some phrases in a semantical consistent way. As shorter emails are later padded to a common length, the identification is even more difficult than simply looking onto the last words. Finally, identification should not be made simpler by biasing the dataset towards senders with many emails. Consequently, among all employees with at least 1000 emails, 1000 emails were chosen randomly. The final corpus now contains $80,000$ emails of 80 different employees for a sequence length of 200 and $90,000$ of 90 different employees for a sequence length of 100.

Further data preprocessing was then conducted in *Python* 3.7.4. All emails were tokenized into words and punctuation marks as well as stopwords removed. Finally, the maximal email lengths were set to 70 and 135, each with a share of 3.3% truncated emails (supposedly without farewell phrase).

3.2 Word Embedding Corpus

Since the anonymization approach will try to remove sender names and characteristic text features, it strongly depends on a word embedding which is trained on all these. It seemed that the pretrained Word2Vec is able to adapt its weights when further trained but will not include new

words. Consequently, there was the need to train an own embedding. The used word embedding is trained on the email corpus itself and several text datasets from kaggle, using Python's *gensim* package. In addition to the Enron Email corpus, several data sets from **kaggle** were used to enrich the corpus to train the embedding on:

- $306,242$ *US Financial News Articles*
- $\approx 35,000$ *Wikipedia Movie Plots*
- $> 180,000$ *Australian Election Tweets from 2019* and
- *A Million News Headlines*

Afterwards, embeddings of different dimensions (50, 60, 80 and 100) were trained.

3.3 Bidirectional LSTM Network for Sender Identification

In order to evaluate the anonymization progress with a base model, a good network for sender identification had to be determined. Since the Enron Email Dataset enjoys great popularity due to its rare size, there have already been various attempts to automatically identify the senders. The bidirectional Long-Short-Term-Memory Network used is taken from [(16)], a github repository of several attempts to identify the senders of Enron emails. The result has the following structure:

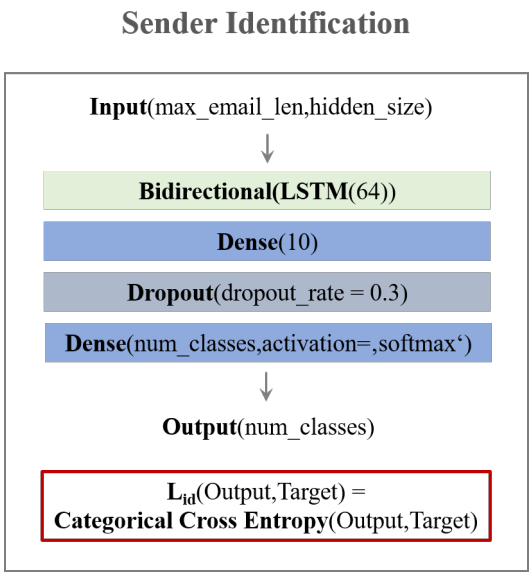


Figure 26: Bidirectional LSTM for Sender Identification

3.4 Sender Identification with Attention

The main idea of the anonymization mechanism presented here, is to add noise to word vectors of those words which are identified to support the sender identification. The idea is further that a network containing an attention mechanism and optimizing a loss function in order to identify senders, will adapt the attention weights towards the respective important parts of the text. A feedforward network with a similar structure as the LSTM is used. An additional dense layer with only two units is introduced to reduce the dimensionality before flattening. The reason for

this is that there are now 2· *hidden_length* ·90 (resp. 80) instead of 20· *hidden_length* ·90 (resp. 80) connections which decreases the computational effort significantly and does in addition not exhaust the information content of the training data in terms of too numerous parameters. The network structure looks as follows:

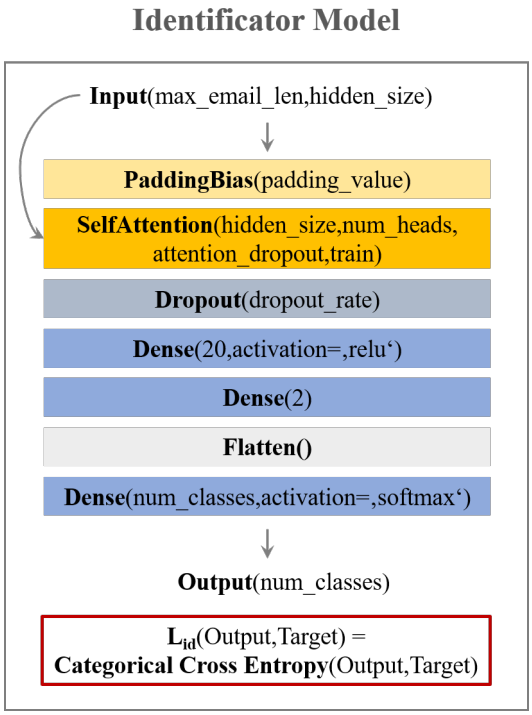


Figure 27: The Identificator Network

The attention mechanism is implemented as a layer in accordance with a former tensorflow implementation of the Transformer model in Github [(37)]. In order to retain position-related information, positional encoding is added element-wisely to the embedded vectors before they are fed to the network. This is the first difference to the original implementation, because there the embedding was trained as part of the model and therefore the positional encoding is added inside the model. In this thesis, the model was completely implemented in keras which unfortunately resulted in a loss of flexibility and further differences in comparison to the original implementation. Instead of Dense Layers to perform the linear projections of the queries, keys and values inside the **SelfAttention** layer, a multiplication with trainable weight matrices is implemented. This amendment was necessary as keras does not seem to support the implementation of layers inside other layers. If the attention weights were adapted to padded values also, they would lose some of their potential. Therefore, a mechanism inside [(37)] sets these weights to zero. This idea is here packed into a **Padding_Bias** Layer. It takes the current input batch and marks padded words with $-\infty$. Inside the attention mechanism in the subsequent SelfAttention layer, a softmax operation is applied on the sum of the weighted input and the padding bias. The corresponding sum is again a matrix with all values corresponding to padded words set to $-\infty$. The softmax operation to calculate the corresponding attention weights allocates the weight 0 to padded words as $\exp(x) \overset{x \to -\infty}{\longrightarrow} 0$. The dropout mechanism proposed in the original Transformer here happens only after the SelfAttention layer as again keras did not allow to use a layer inside a layer. Abbreviations from the results presented in

[(3)] are expected in consequence.

To assess the improvement coming along with self-attention in comparison to a simple feed forward network, the network of the Identificator without Padding_Bias and SelfAttention layer is evaluated also. The corresponding network will be called Basic Feedforward Network (Basic FFN). Its structure is displayed below.

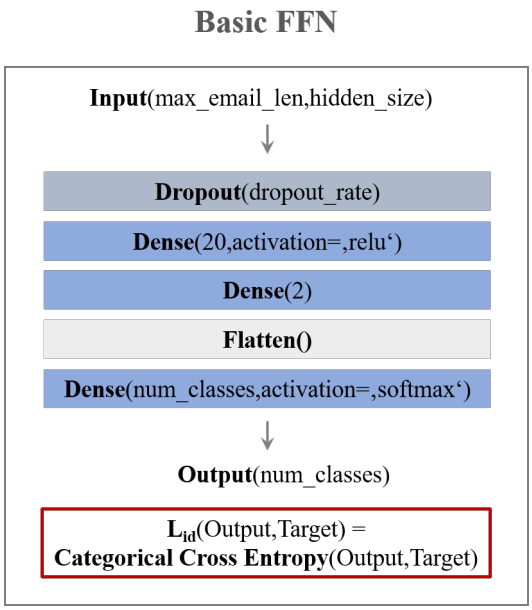


Figure 28: The Basic FFN - Without Attention Mechanism

3.5 Combine Attention and a Privacy Mechanism

The privacy mechanism proposed in [(21)] makes use of a multivariate Laplacian because of its appealing form (most of the noise lies within a small surrounding). In the corresponding paper, it was firstly important to show that it is possible to transfer $d_{\mathcal{X}}$-privacy from vectors to matrices (from a row of data to a bag of words each represented as a word vector). The mechanism adds a multivariate Laplace noise vector to each word vector independently. Unfortunately, there was so far no implementation of the multivariate Laplacian and consequently the authors had to derive a solution to this problem first. However in this thesis, because the independence of all noise should not influence the model outcome in a negative way, no multivariate Laplacian but a matrix of Laplace random numbers is used. A $d_{\mathcal{X}}$-private algorithm making use of this Laplace Matrix is proposed. Instead of word-wise addition, the noise is added to each entry of the document matrix in an independent manner. The replacement of a multivariate Laplacian with i.i.d. random Laplace numbers also simplified the proof of privacy for the proposed mechanism. Still, the mechanism had to be extended from one entry to the whole document matrix.

The below mechanism together with the information regarding the Manhattan Metric is based on [(21)] but was adapted in order to involve the multiplication with the attention matrix. Because the attention matrix has the same size as the document, the mechanism is formulated in terms of one document instead of one (word)vector.

Theorem 3. (Laplace mechanism for documents with Attention)
Given $\epsilon > 0$ and $n, m \in \mathbb{Z}^+$, let $\mathcal{M}_{\text{Att}} : \mathbb{R}^n \to \mathbb{DR}^{n \times m}$ be a mechanism that, given a matrix $X \in \mathbb{R}^{n \times m}$ and a matrix $A \in \mathbb{R}_+^{n \times m}$ outputs a noisy value as

$$X \mapsto X + A \cdot X',$$

where $X' \in \mathbb{R}^{n \times m}$ consists of i.i.d. random numbers distributed as $\text{Lap}(\frac{1}{\epsilon})$. The product of X' and A is conducted element-wise. Then $\mathcal{M}_{\text{Att}}$ satisfies $\frac{\epsilon}{\min\limits_{i,j} |a_{ij}|}||\cdot||$-privacy where $||\cdot||$ denotes the so-called Manhattan Metric computed as the sum of the element-wisely computed L_1-norm on $\mathbb{R}^n$.

Proof. The according proof is part of the appendix. □

As in this master thesis the entries of A do not correspond to hyperparameters, ϵ remains the only hyperparameter in charge of the privacy-utility-trade-off.

The following algorithm represents an adaptation of the in [(21)] proposed algorithm to apply the concept of Earth-Mover's-Privacy to bags-of-words. It is adapted in several ways. On the one hand, emails are not represented as bag-of-words and the term "bag" is replaced with "email" in consequence. On the other hand, the noise is computed as a matrix in a compact form and then multiplied with attention values.

Algorithm Earth Mover's Privacy Mechanism with Attention

Require: document X, hidden_size m, max_email_len n, epsilon ϵ
1: **procedure** *GenerateNoisyMatrix*(X, n, m, ϵ)
2: Sample $Z \in \mathbb{R}^{n \cdot m} \overset{\text{i.i.d}}{\sim} \text{Lap}(\frac{1}{\epsilon})\ n \cdot m$
3: Reshape Z as (n, m)
4: **return** Z
5: **end procedure**
Require: mail X, attention matrix A, hidden_size n, epsilon ϵ
1: **procedure** *GeneratePrivateMail*(X, A, n, m, ϵ)
2: Z $\leftarrow$ *GenerateNoisyMatrix*(X,n,m,ϵ)
3: P $\leftarrow$ $X + A \cdot Z$
4: **return** P
5: **end procedure**

Subsequently to training the attention weights with respect to author identification, those weights are extracted in form of a *max_email_len* $\times$ *hidden_size* matrix W_{Att}. When extracted, W_{Att} has values in a different range than $[0, 1]$, as no softmax was applied after the last linear projection. However, in order to achieve the desired effect, all matrix entries must be scaled to this interval and therefore a softmax is applied:

$$A = \text{softmax}(W_{\text{Att}}) \tag{59}$$

In the following, this matrix A is referred to as attention matrix. Next, A is multiplied element-wisely with an equally sized matrix of Laplace noise, generated as described above. Higher weights should then result in higher noise values as

$$\frac{\epsilon}{a_{ij}} \xrightarrow{a_{ij} \to 1} \epsilon \tag{60}$$

which fits the purpose, as to more significant words more noise should be added to. In contrast,

$$\frac{\epsilon}{a_{ij}} \xrightarrow{a_{ij} \to 0} \infty, \tag{61}$$

accounts to less important words and at this scale nearly no noise is produced under a Laplace distribution. In a final step, the weighted noise matrix is added to the input text.

In none of these definitions the Earth Mover's Metric was used. The next section about the measurement of the remaining data utility provides a more extensive derivation of reasons, at this point it is sufficient to know that the Earth Mover's Metric for documents of equal length differs from the specific form of Manhattan Metric used only by the constant factor $\frac{1}{max\_email\_len}$ which is irrelevant for optimization. The above theorem therefore also implies Earth Mover's Privacy.

From the explanations before, the larger ϵ the less privacy is created but the more data utility preserved. However, thanks to the multiplication with the inverse of the scaled attention values the more sensitive the data point x_{ij}, the larger is the corresponding value a_{ij} and the more privacy is produced and vice versa. The definition profits furthermore from the softmax, as A does not have zero-valued entries and there is no unfavourable division with 0 in $\frac{\epsilon}{a_{ij}}$. However, the effect of A on ϵ is two-fold, as on the one hand the privacy guarantee depends on the ability of W_{att} to capture all sensitive data and on the other hand, the privacy guarantee is only a lower bound to a locally varying bound.

3.6 Measure of Remaining Data Quality

On Github, several unsupervised tasks like sentiment analysis and spam detection were executed on the Enron Email Data. Both options to determine the data utlity were excluded here for two reasons. On the one hand, spam emails were deleted during the data cleansing process in order to train attention as accurately as possible, and on the other hand the email quality was not sufficient to perform a task with a good accuracy from the start. In [(21)] not only the level or privacy but also the data utility was evaluated with a measure of topic relatedness - the Earth Mover's Distance. The idea is that the distance d "is a measure of utility" [(21), p.14] itself, as further tasks will need the output text to be close in topic to the input text. Consequently, it seems reasonable to adopt this idea and select the utility loss L_{utility} as the Earth Mover's Distance between input and its anonymized equivalent.

The corresponding loss makes then use of either the euclidean or the L_1-norm to evaluate the distance between words at the same positions in two documents. All other words are neglected

as gets obvious from the equations below. The Earth Mover's Metric corresponds to

$$E_{d_S}(X, Y) := \sum_{x_i \in X} \sum_{y_j \in Y} d_S(x_i, y_j) F_{ij} \tag{62}$$

where the flow matrix is of the form

$$F_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j. \end{cases} \tag{63}$$

if both emails are of the same size. Consequently, the definition reduces to

$$E_{d_S}(X, Y) := \sum_{i=1}^{\text{max\_email\_len}} d_S(x_i, y_i) F_{ii} = \frac{1}{max\_email\_len} \sum_{i=1}^{\text{max\_email\_len}} d_S(x_i, y_i). \tag{64}$$

In terms of optimization the factor $\frac{1}{max\_email\_len}$ may be neglected as constant factors do not influence the optimization process. If the factor is not neglected the term above simply corresponds to the mean distance of words between an original email and its replacement. The EMD loss was implemented in keras as the custom loss *emd_loss*.

3.7 Combine Privacy and Utility Trade-Off in an Adversarial Loss

In order to involve the privacy-utility trade-off in the training process, it seemed a valuable attempt to combine the contradictory goals inside an adversarial loss. While the anonymization mechanism, a measure of utility and a model to train the attention values with respect to sender identification are already described, a strategy to combine all into one model is missing at this point. The figure below illustrates at which point W_{att} is extracted. It is contained in the output

$$\text{attention\_output} = inputs + W_{\text{att}} \tag{65}$$

of the SelfAttention layer inside the Identificator network. The attention output is then fed to the **Noiser** model. In the custom layer **AttnTens**, the input is simply subtracted in order to obtain W_{att}. The anonymization mechanism is then applied inside the custom **LaplaceNoise** layer with respect to privacy hyperparameter ϵ. In order to support the attention matrix inside the Identificator in a way such that it focuses on the sensitive data only, the topic-relatedness of input email and anonymized email should be maximized and the Earth Mover's Distance between these emails minimized as a consequence. The above described *emd_loss* induced by the euclidean norm $|| \cdot ||_2$ is used as L_{utility}. The EMD loss induced by the L_1-norm $|| \cdot ||_1$ is discarded as in gradient-based optimization the model does not learn with respect to a non-concave function. The model implementation is strongly oriented in [(25)].

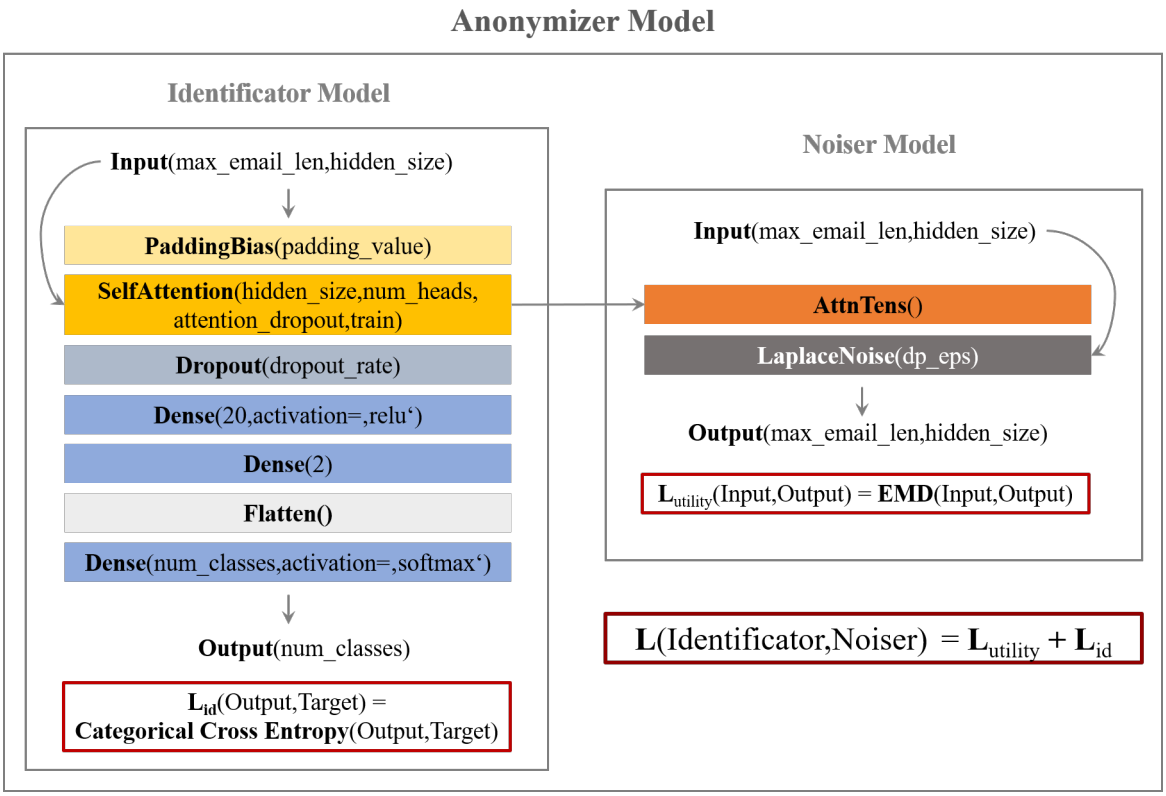


Figure 29: Adversarial Loss: Privacy vs. Utility

The training procedure reveals that the Anonymizer incorporates an adversarial loss but not the one distinct to GANs. Instead, W_{att} is updated twice - by alternating training with respect to the sender identification and training in order to decrease the EMD.

3.8 Evaluation of Performance

For all models email lengths of 200 (135) and 100 (70) and embeddings of different sizes (50, 60, 80, 100) were compared. The maximal accuracy to obtain because of the information contained in the appended farewell phrase is decreased to ≈ 0.97 because a small fraction of emails had more words than the specified threshold of 70 respectively 135. Finally, *hidden_size*=60 was selected for emails containing 100 words and *hidden_size*=100 for emails with 200 words. The corresponding comparison of learning curves is contained in the appendix. Furthermore, several values for the differential privacy hyperparameter ϵ were compared to each other. The table below shows the separation of the respective data sets into train, validation and test data and the parameter setting in terms of training.

max_email_len	Train	Validation	Test	Total
100 (70)	44100	18900	27000	90000
200 (135)	39200	16800	24000	80000

Table 2: Train/Validation/Test Split $(0.49, 0.21, 0.3)$

max_email_len	batch_size	steps_per_epoch	Total steps
100 (70)	70	630	12600
200 (135)	70	560	11200

Table 3: Training parameters

3.8.1 Sender Identification

The bidirectional LSTM seems to be very well able to extract the information supporting the identification of senders. It seems a little surprising that the validation accuracy is almost twice as high as the training accuracy. This is partly caused by the dropout layer but still a lot more expressive than expected. However, with a value of 0.937 the test accuracy corresponds to the validation accuracy and consequently the result seems to be correct.

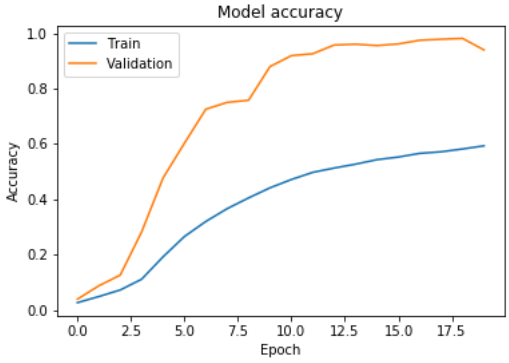
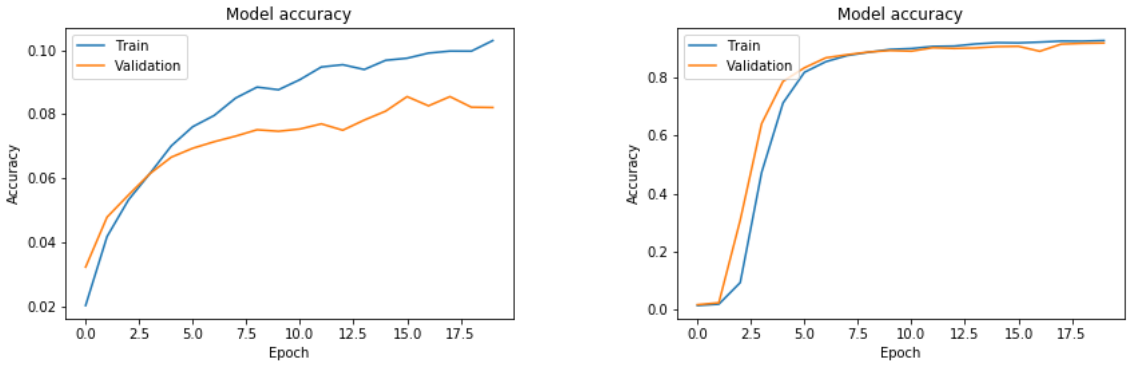


Figure 30: Training and Validation Accuracy of Bidirectional LSTM

3.8.2 Basic FFN Model in Comparison to the Identificator

The difference in performance between the Basic FFN and the Identificator is quite remarkable. The Basic FFN is capable of escaping the accuracy baseline of random guessing but is stuck at a validation accuracy below 0.10. The corresponding test error amounts to 0.1078 and therefor underlines the result. In contrast, the Identificator shows a rather steep learning curve with a diminishing gap between training and validation error. With a value of 0.9173 the test error supports the validation accuracy displayed. It can also be stated that both test accuracies, those of the Identificator and the Bidirectional LSTM, belong to a similar range.



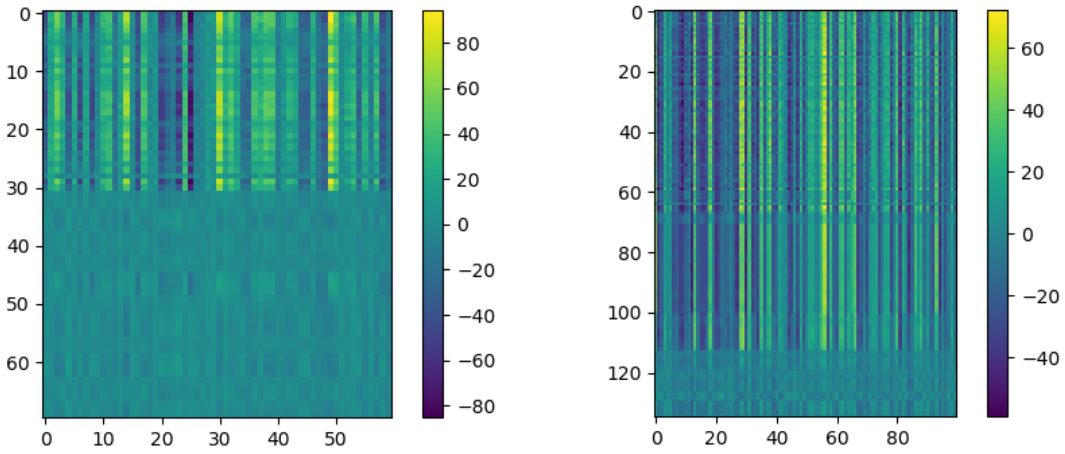
(a) Basic FFN

(b) Identificator model

Figure 31: Overview of Training and Validation Accuracy of the Basic FFN

3.8.3 Anonymization by Hand

To compare the results of the adversarial loss with W_{Att} trained with respect to sender identification (L_{id}) only, the anonymization mechanism was also applied by hand. An intermediate model was built from the before trained Identificator and the corresponding W_{Att} extracted. Then, a matrix of i.i.d. Laplace noise was multiplied element-wisely with the attention matrix and the result transformed back from vectors into words. Last, the distance of input and anonymized output and the text quality were assessed.
Below, W_{Att} is visualized examplarily for both *max_email_len* values.

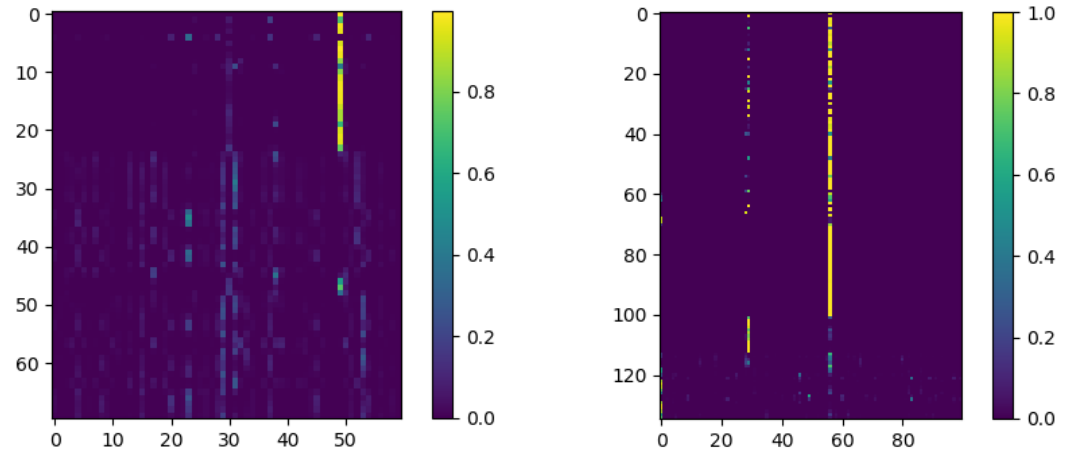


(a) *max_email_len*=100 (70)

(b) *max_email_len*=200 (135)

Figure 32: Weight Distribution in W_{Att}

The visualization of W_{Att} illustrates the effect of the padding bias very well: All padded words have zero weight. However, it gets furthermore obvious why the anonymization mechanism will not work as supposed if this output is used as A. The weights are neither positive nor less than one. The scaled matrix values after the subsequent softmax application are displayed below.



(a) *max_email_len*=100 (70)

(b) *max_email_len*=200 (135)

Figure 33: Extracted Attention Output After Softmax Application

It is quite obvious that now specific parts are emphasized. However it is surprising that the focus lies upon a specific position in each embedding vector and not on specific parts of the

email. On the other hand, the information was at a different position in most emails. Next, random search is performed in order to find a suitable ϵ. It seemed useful to compare the values 0.03, 0.001, 0.0001 and 0.00002 for both settings. Below, one original email with its respective anonymized version can be seen.

Original	**Anonymous**
['calendar', 'entry:', 'appointment', 'description:', 'gpg', 'united', 'way', 'rally', '-', 'energizer', 'date:', 'adjustability', 'time:', '3:00', 'pm', '-', '4:00', 'pm', 'central', 'standard', 'time', 'detailed', 'description:', 'would', 'grateful', 'would', 'get', 'touch', 'blair-l', 'soon', 'possible']	['663997', 'entry:', 'bopper"', 'description:', 'knockinâ\x80\x99', 'united', 'way', 'rally', '-', 'onwsjcom/2hscwhs', 'date:', 'knockinâ\x80\x99', 'time:', '12:00', 'pm', '-', '4:00', 'pm', 'central', '663997', 'time', 'detailed', 'description:', 'would', 'grateful', 'would', 'get', 'touch', 'bitly/2f6983y', 'soon', 'possible']

Table 4: Original and Output after Application of Anonymization Mechanism (Padding Removed), $\epsilon = 0.03$, *max_email_len*=100

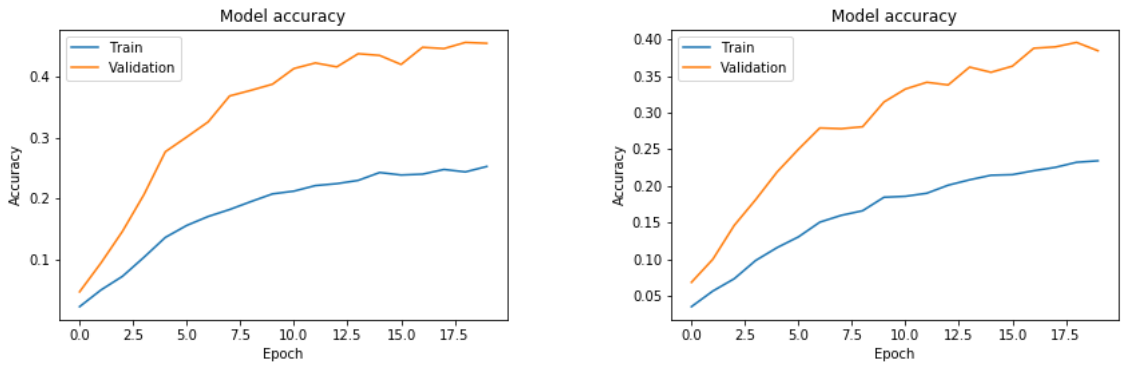
In this example it looks like it has worked quite well. The name in the farewell phrase is replaced but many other words are preserved. It is also worth noting that a time has been replaced by another time. In a different example, the sender name was even replaced with the name of a different person. The corresponding example is displayed below.

Original	Anonymous
['got', 'package', 'fed', 'ex', 'thought', 'would', 'let', 'know', 'thanx', 'chris', 'faithfully', 'dorland-c']	['84508', 'package', 'fed', '<u+200d>senvest', 'thought', 'would', 'let', 'schillings', 'iosadmin', 'chris', 'blackout', 'rodrique-r']

Table 5: Original and Output after Application of Anonymization Mechanism 2 (Padding Removed), $\epsilon = 0.03$, *max_email_len*=100

A replacement with the same word class is supposed to keep the data utility as high as possible. This effect was desired from the beginning and reflected in the decision on a trained embedding.

However, the LSTM still easily identifies the senders. With $\epsilon = 0.00002$ the LSTM finally struggles to identify senders in both settings.



(a) *max_email_len*=100 **(b)** *max_email_len* =100

Figure 34: Learning Curve of the Bidirectional LSTM on Anonymized Data ($\epsilon = 0.00002$).

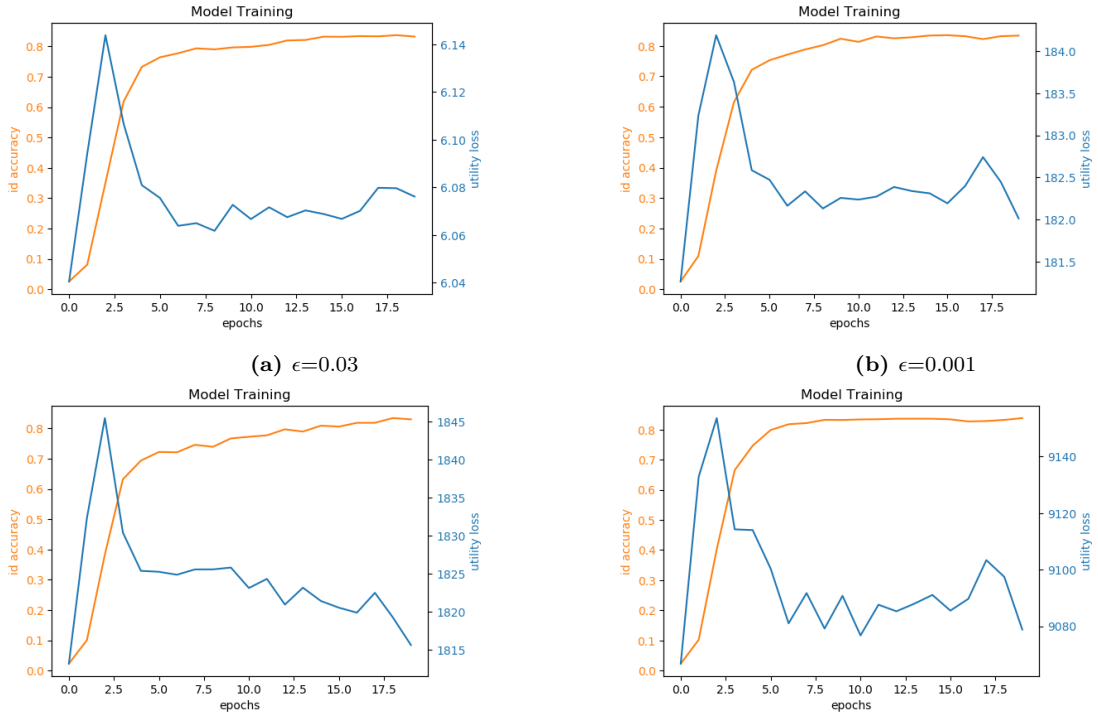
Unfortunately, the utility suffers in a too serious manner from this level of privacy and all words are replaced. It should additionally be remarked that only half as much data was available for the LSTM training in comparison to the earlier Identificator training, because test data was used in order to prevent a possible profit from any overfitting to training data. Still, with an accuracy of less than 50%, it is possible to say that at least 2-anonymity is secured in that the LSTM can no longer differ between two different senders. But as noted above, this anonymity is at the cost of any visible data utility because the anonymized text does not contain many of the previous words and no structure at all.

The training with the adversarial loss may improve the outcome.

3.8.4 Results of the Adversarial Loss Approach

In the adversarial setting, the embedding with size 60 was again used for *max_email_len*=70, and *hidden_size*=100 in the case where *max_email_len*=135. The results of both settings lead to similar conclusions and therefore only the results of the 60/70 setting are described here. All corresponding results of the 100/135 setting are part of the appendix.

The combined model does not learn remarkably but the loss decreases at least in a little range which may already indicate an improvement in comparison to the manual approach.



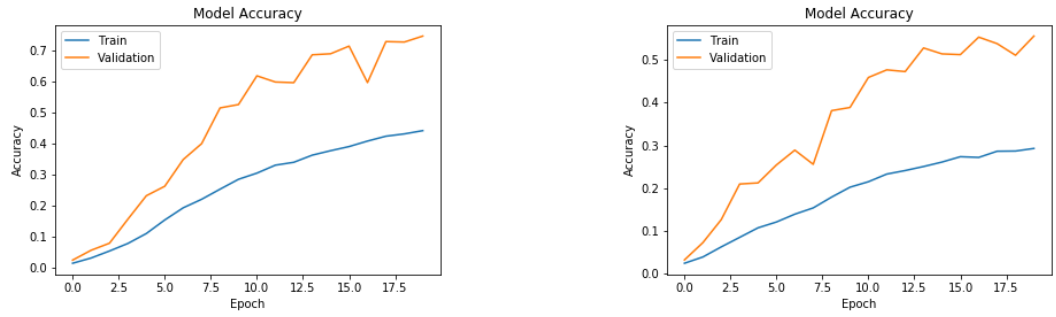
(a) ϵ=0.03 **(b)** ϵ=0.001

(c) ϵ=0.0001 **(d)** ϵ=0.00002

Figure 35: Learning Curve of the Anonymizer (*max_email_len*=70)

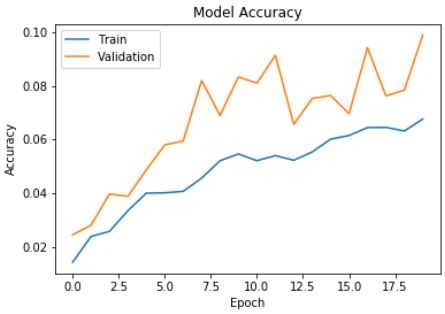
The Anonymizer model was then fit to test data and the output again fed to the LSTM to assess the success of sender identification as a labelled task on the anonymized data.

It looks like the Anonymizer achieves a decrease of the LSTM's predictive performance a lot faster than the manual approach. At $\epsilon = 0.001$, after 20 epochs of training it looks like the LSTM has already difficulties in distinguishing between 2 different senders and at $\epsilon = 0.0001$ 10-anonymity is achieved. In the 100/135 setting, 12-anonymity is even reached at $\epsilon = 0.001$ already. However, the really interesting question to be answered is whether the adversarial loss approach was better capable of preserving the text utility at the same privacy hyperparameter ϵ. The corresponding learning curves are displayed below in fig. 36.

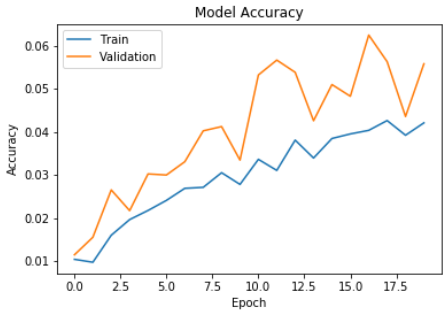


(a) ϵ=0.03

(b) ϵ=0.001



(c) ϵ=0.0001



(d) ϵ=0.00002

Figure 36: Learning Curve of the LSTM on Anonymized data in the Adversarial Setting (*max_email_len*=70)

3.8.5 Comparison in Privacy-Utility-Trade-Off

The utility loss for both models under the same parameter setting is displayed below for *max_email_len*=70.

ϵ	**Manual**	**Anonymizer**
0.03	23.76	6.14
0.001	768.47	180.08
0.0001	7793	1815
0.00002	41516	9080

Table 6: Comparison of Utility Loss, Manual vs. Adversarial Approach, *max_email_len*=70

The adversarial model indeed seems to preserve a remarkable higher degree of data utility. The remaining utility is however questionable as almost none of the previous words are preserved. In comparison to the manual approach at $\epsilon = 0.001$, the replacements still appear similar messy.

Original	Manual	Adversarial
['euro-plus',	['knockinâ\x80\x99',	['https://tco/nbxqzen6eb',
'ajilon',	'knockinâ\x80\x99',	'goggia\x92s',
'please',	'onwsjcom/2hscwhs',	'tested\x94',
'note',	'blue-tick',	'pop-band',
'columbia',	'knockinâ\x80\x99',	'nt$25991',
'navigator',	'knockinâ\x80\x99',	'outjumped',
'system',	"'mining'",	'ulaanbaatar',
'unavailable',	'unavailable',	'foldout',
'every',	'knockinâ\x80\x99',	'wwwh2oai',
'sunday',	'sieg',	'saab',
'11:00',	'pools:',	'??-?great',
'pm',	'onwsjcom/2hscwhs',	'non-borrowing',
'eastern',	'onwsjcom/2hscwhs',	'lassman',
'monday',	'knockinâ\x80\x99',	'cabret\x94',
'6:00',	'25th',	'wwwfuturesindustryorg',
'eastern',	'onwsjcom/2hscwhs',	'three-pointers',
'routine',	'knockinâ\x80\x99',	'whistling[1]',
'system',	'onwsjcom/2hscwhs',	'counter-cyclical',
'maintenance',	'pools:',	'https://tco/owqvcmnu73',
'kind',	'onwsjcom/2hscwhs',	'rpt-analysis-what',
'regards',	"'mining'",	'15166254',
'hodge-j']	'onwsjcom/2hscwhs',]	'bitly/2fzmcw6']

Table 7: Example Comparison of Original and Anonymized Emails (*max_email_len*=70)

However, for *max_email_len*=135, more utility is kept in the adversarial approach at each ϵ compared to the adversarial approach with *max_email_len*=70. In contrast, the utility loss of the manual approach increased with email length. The corresponding privacy-utility trade-off can be seen from the below table.

ϵ	**Manual**	**Anonymizer**
0.03	32.70	4.70
0.001	929.73	140.7
0.0001	9045	1410
0.00002	42890	7050

Table 8: Comparison of Utility Loss, Manual vs. Adversarial Approach, *max_email_len*=135

A reason for this behaviour is not evident. A possible reason could be the larger embedding size that serves to capture even more contextual characteristics. The anonymized emails however look similar distorted than for *max_email_len*=70 and also in all other respects the same conclusions can be drawn from the results.

4 Summary and Discussion

First, a suitable data set for anonymization was searched for and found in the open accessible Enron Email Data. An individual word embedding was trained in order to guarantee that all sensitive words like names are contained and will keep their contextual similarities inside the embedding. All emails were preprocessed in order to satisfy a standard structure. For later comparison of predictive power between an original dataset and its anonymized equivalence, a sender identification network was developed as a bidirectional LSTM. A second network for sender identification, the so-called Identificator, was constructed but incorporates an attention mechanism instead of recurrency. In order to evaluate the effect of the attention mechanism, a third model corresponding to the Identificator without attention, was built for additional performance comparison. Last, a model combining the Identificator with a second network, the Noiser network that incorporates the anonymization mechanism, was developed. The corresponding model is called the Anonymizer and enables automation of the anonymization process. The automation works thanks to an adversarial loss function involving a privacy-utility trade-off.
Initially, the anonymization mechanism was performed manually to enable evaluation of the automated approach. In contrast to the adversarial approach, the privacy-utility trade-off does not participate in the training process when the noise addition is applied by hand. The anonymized outputs of both approaches were further examined in terms of remaining predictive power of the bidirectional LSTM for sender identification.
While the manual attempt produced the intuitively desired effect of anonymization of as obviously sensitive regarded data only for larger privacy budgets, too much data utility was lost when using a small privacy budget that achieves to prevent re-identification through a bidirectional LSTM. The model incorporating an adversarial loss was in each setting able to reduce the initial utility loss but only in a small range. A great deal of data utility was lost here, too, before the LSTM could no longer clearly identify the sender. However, in comparison to the manual approach a remarkably higher degree of data utility was preserved and the LSTM was already confused at a larger privacy budget.
Still, this suggests that if larger privacy budgets are used, not all sender names are exchanged in the email text. In addition, the LSTM may in some cases identify the senders not only by a name that can be replaced, but also by other semantic characteristics like word combinations. However, it should be stressed that this investigation is not realistic in that an adversary will usually not have access to labelled data. In contrast, an unsupervised setting is far more likely. Furthermore, the embedding's ability to reflect the context may support the LSTM in a way such that words close to the sender name and not the name itself are sufficient for re-identification. The desired effect of the trained embedding, namely the enablement of word vector replacements with word vectors in their spatial proximity that are also close in a semantic sense, may also influence the privacy in an unfavorable manner. A further step could therefore be the use of an embedding that is trained with respect to a privacy guarantee as in [(13)]. Finally, the randomness in differential privacy is both a curse and a blessing. On the one hand, privacy is introduced but on the other hand randomness diminishes the ability of control. As a consequence, sensitive attributes may be allocated little or zero noise to at random. Cryptographic terms of privacy pose an alternative to differential privacy and are e.g. proposed as part of a GAN in [(2)].

In conclusion, it can be said that the goal of the work was achieved to the extent that, on the one hand, a mechanism was found that combines attention and differential privacy. Furthermore, there were indeed settings in which utility was preserved as the replacements matched the class of the exchanged words. On the other hand, a way was found to let the privacy-utility trade-off actively participate in the training and thus automate the anonymization. However, there is room for improvement in the form of a better embedding, a different anonymization mechanism and a more successful adversarial training.

List of Figures

List of Tables

Bibliography

[1] Cynthia Dwork and Aaron Roth. *The Algorithmic Foundations of Differential Privacy.* https://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf, Foundations and Trends in Theoretical Computer Science Vol. 9, Nos. 3–4 (2014) 9 p. 211–407, 2014.

[2] Ho Bae, Dahuin Jung and Sungroh Yoon. *AnomiGAN: Generative adversarial networks for anonymizing private medical data.* https://arxiv.org/abs/1901.11313, published Jan 2019.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser and Illia Polosukhin. *Attention Is All You Need.* https://arxiv.org/abs/1706.03762, published Dec 2017.

[4] Ravindra Parmar. *Common Loss Functions in Machine Learning.* https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23 accessed Jan 13 2020.

[5] Prodip Hore and Sayan Chatterjee. *A Comprehensive Guide to Attention Mechanism in Deep Learning for Everyone.* https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism- accessed Jan 20, 2020.

[6] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf, Cambridge University Press, 2004.

[7] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning.* MIT Press, https://www.deeplearningbook.org, published 2016.

[8] Prof. Dr. Bernd Bischl and Janek Thomas. *Deep Learning 18/19.* Lecture, winter term 2018/2019.

[9] Ismini Psychoula, Erinc Merdivan, Deepika Singh, Liming Chen, Sten Hanke, Johannes Kropf, Andreas Holzinger and Matthieu Geist. *A Deep Learning Approach for Privacy Preservation in Assisted Living.* https://arxiv.org/abs/1802.09359, published Feb 2018.

[10] Martin Abadi, H.Brendan McMahan, Andy Chu, Ilya Mirononv, Li Zhang, Ian Goodfellow and Kunal Talwar. *Deep Learning with Differential Privacy.* https://arxiv.org/pdf/1607.00133.pdf, published Oct 2016.

[11] Kyunghyun Cho, Aaron Courville and Yoshua Bengio. *Describing Multimedia Content using Attention-based Encoder-Decoder Networks.* https://arxiv.org/pdf/1507.01053.pdf, published Sep 2015.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado and Jeffrey Dean. *Distributed Representations of Words and Phrases and their Compositionality.* https://arxiv.org/abs/1310.4546, published Oct 2013.

[13] Xuan-Son Vu, Son N. Tran and Lili Jiang. *dpUGC: Learn Differentially Private Representation for User Generated Contents.*
https://arxiv.org/abs/1903.10453, published Mar 2019.

[14] Minh-Thang Luong, Hieu Pham and Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation.*
https://arxiv.org/abs/1508.04025, published Sep 2015.

[15] Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space.*
https://arxiv.org/abs/1301.3781, published Sep 2013.

[16] Vighnesh Chenthil Kumar, Kritika Prakash, Aditiya Srivastava and Karthik Chintapalli. *E-Mail Author Identification.*
https://github.com/vighneshck/E-mail-Author-Identification/blob/master/models/LSTM_final.py, accessed Oct 2020.

[17] Rachel Smith. *The Enron Scandal.*
http://large.stanford.edu/courses/2018/ph240/smith1/, accessed Nov 2019.

[18] William W. Cohen. *Enron Email Dataset.*
https://www.cs.cmu.edu/~./enron/, accessed Jul 24 2019.

[19] David Marker. *Game Theory, spring 2013.*
http://homepages.math.uic.edu/~marker/stat473-S13/zero-sum.pdf, ch. 1, University of Chicago, accessed Jan 20, 2020.

[20] https://gdpr-info.eu/, accessed Jan 25, 2020.

[21] Natasha Fernandes, Mark Dras and Annabelle McIver. *Generalised Differential Privacy for Text Document Processing.*
https://arxiv.org/abs/1811.10256, published Feb 2019.

[22] Anonymous authors. (under double blind review) *Generating Differentially Private Datasets Using GANs.*
https://openreview.net/pdf?id=rJv4XWZA-, unknown.

[23] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Wade-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. *Generative Adversarial Nets.*
https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf, published Jan 2014.

[24] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta and Anil A. Bharath. *Generative Adversarial Networks: An Overview.*
https://arxiv.org/abs/1710.07035, published Oct 2017.

[25] Sebastian Theiler. *Implementing a GAN in Keras*
https://medium.com/analytics-vidhya/implementing-a-gan-in-keras-d6c36bc6ab5f, accessed Dec 3rd, 2019.

[26] Jason Brownlee. *A Gentle Introduction to Dropout for Regularizing Neural Networks.* https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/, accessed Jan 26, 2020.

[27] Clément Feutry, Pablo Piantanida, Yoshua Bengio and Pierre Duhamel. *Learning Anonymized Representations with Adversarial Neural Networks.* https://arxiv.org/pdf/1802.09386.pdf, published Feb 2018.

[28] Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro and Hamed Haddadi. *Mobile Sensor Data Anonymization..* https://arxiv.org/abs/1810.11546, published Feb 2019.

[29] Dzmitry Bahdanau, KyungHyun Cho and Yoshua Bengio. *Neural Machine Translation By Jointly Learning To Align And Translate.* https://arxiv.org/pdf/1409.0473.pdf, published May 2016.

[30] Simon Haykin. *Neural Networks and Learning Machines.* Third Edition, Pearson Education Inc., 2009.

[31] Daniel Jurafsky and James H. Martin. *N-Gram Language Models.* https://web.stanford.edu/~jurafsky/slp3/3.pdf, published Oct 2019.

[32] Parameswaran Kamalaruban, Victor Perrier, Hassan Jameel Asghar, and Mohamed Ali Kaafar. *Not All Attributes are Created Equal: d$\mathcal{X}$-Private Mechanisms for Linear Queries.* https://arxiv.org/pdf/1806.02389.pdf, published Aug 2019.

[33] Posted by Jacob Devlin and Ming-Wei Chang. *Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing.* https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html, accessed 11 Jan, 2020.

[34] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer Science+Business Media, 2006.

[35] Kelvin Xu, Jimmy Lei Ba, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel and Yoshua Bengio. *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.* https://arxiv.org/pdf/1502.03044.pdf, published Apr 2016.

[36] Janet Chen, Su-I Lu, and Dan Vekhter. *Strategies of Play.* https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/game-theory/Minimax.html, accessed Jan 20, 2020.

[37] *Transformer model for language understanding.* https://www.tensorflow.org/tutorials/text/transformer, accessed Oct 3 2019.

[38] Christopher Olah. *Understanding LSTM networks.* https://colah.github.io/posts/2015-08-Understanding-LSTMs/, accessed Jan 18, 2020.

[39] Aditi Mittal. *Understanding RNN and LSTM.* https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e, accessed Jan 18, 2020.

[40] David Phelan. *Amazon Admits Listening To Alexa Conversations: Why It Matters.* https://www.forbes.com/sites/davidphelan/2019/04/12/amazon-confirms-staff-listen-to-alexa-conversations-heres-all-you-need-to-know/#3a756c95d9db, accessed Jan 27, 2019.

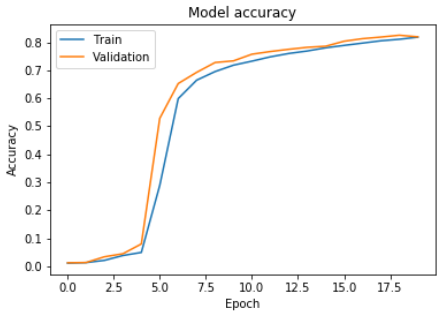
[41] https://www.spektrum.de/lexikon/physik/metrik/9686, accessed Dec 18, 2019

[42] Simon Hurtz. *Warum "anonyme" Daten eine Illusion sind.* (deutsch) https://www.sueddeutsche.de/digital/anonyme-daten-studie-1.4542458, accessed Jul 13, 2019.

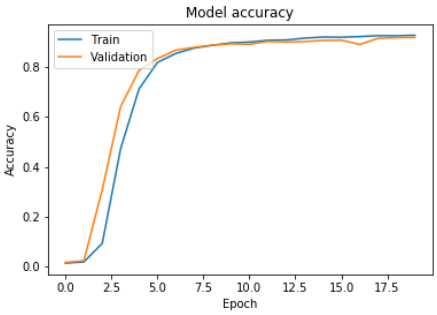
Appendix

A Comparison of different *hidden_sizes*

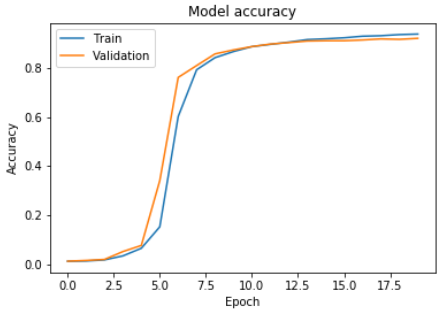
Below are the learning rates displayed to underline the decision for the selected *hidden_size*.



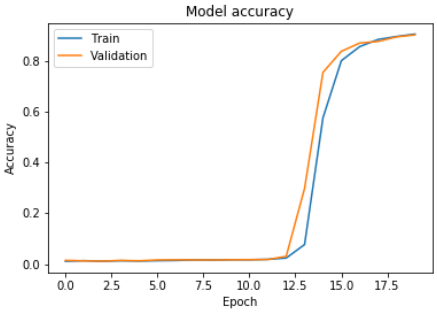
(a) *hidden_size*=50



(b) *hidden_size*=60

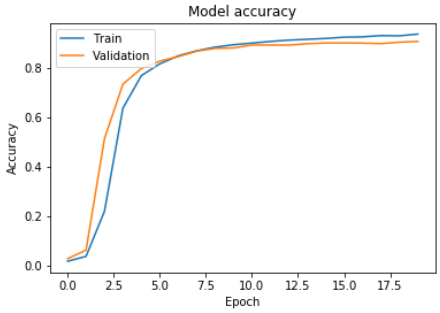


(c) *hidden_size*=80

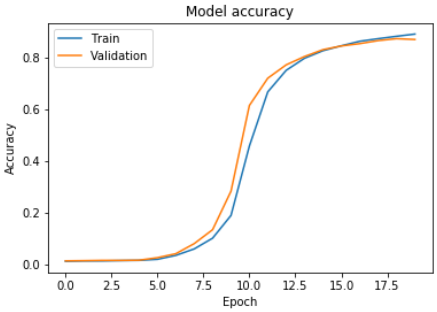


(d) *hidden_size*=100

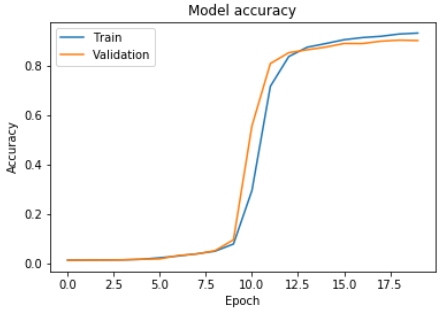
Figure 37: Identificator Training with *max_email_len*=100 (70)



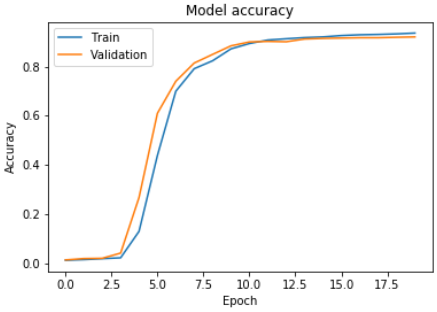
(a) *hidden_size*=50



(b) *hidden_size*=60



(c) *hidden_size*=80



(d) *hidden_size*=100

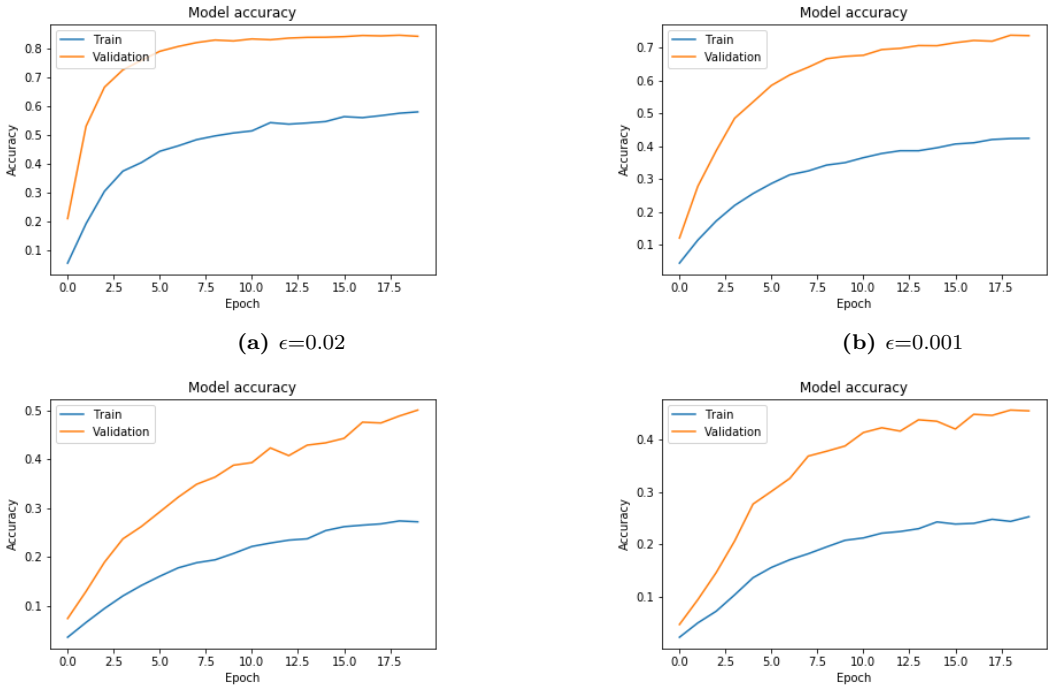
Figure 38: Identificator Training with *max_email_len*=200 (135)

hidden_size	Model	length 100	length 200
50	LSTM	0.9733	0.9555
	Base FFN	0.0869	0.1001
	Identificator	0.8162	0.9028
60	LSTM	0.9721	0.9686
	Base FFN	0.0940	0.1078
	Identificator	0.9173	0.8668
80	LSTM	0.9732	0.9699
	Base FFN	0.1185	0.1251
	Identificator	0.9251	0.9009
100	LSTM	0.9755	0.9657
	Base FFN	0.1282	0.1183
	Identificator	0.9013	0.9224

Table 9: Comparison of Test Accuracies under Different *hidden_sizes* (Chosen Settings Marked in Yellow)

B Predictive Power of Bidirectional LSTM on Anonymized Data

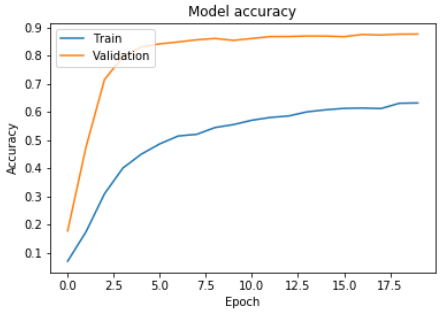
B.1 Manual Procedure



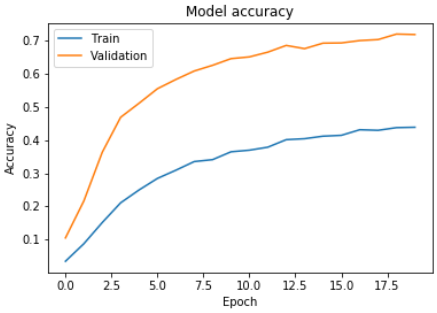
(a) ϵ=0.02 **(b)** ϵ=0.001

(c) ϵ=0.0001 **(d)** ϵ=0.00002

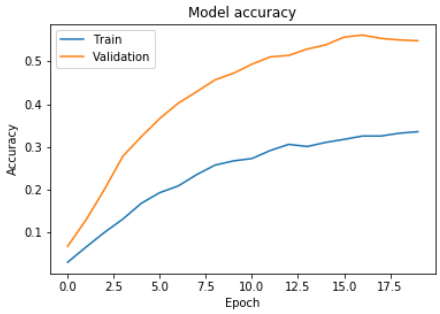
Figure 39: Learning Curve of the LSTM on Anonymized Data (*max_email_len*=70)



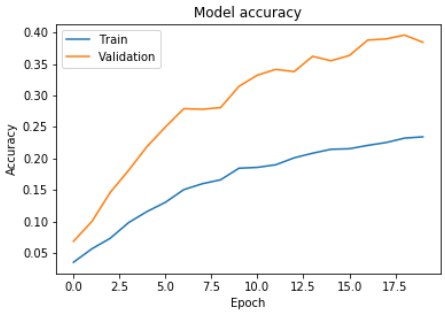
(a) ϵ=0.02



(b) ϵ=0.001



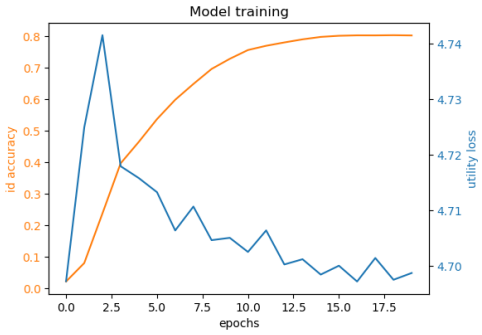
(c) ϵ=0.0001



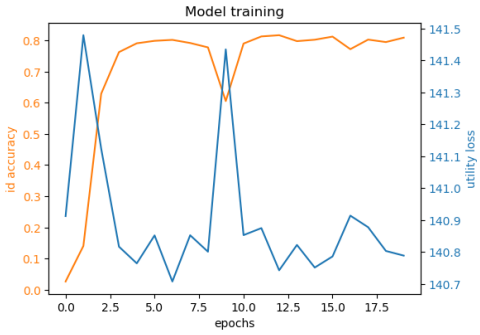
(d) ϵ=0.00002

Figure 40: Learning Curve of the LSTM on Anonymized Data (*max_email_len*=200)

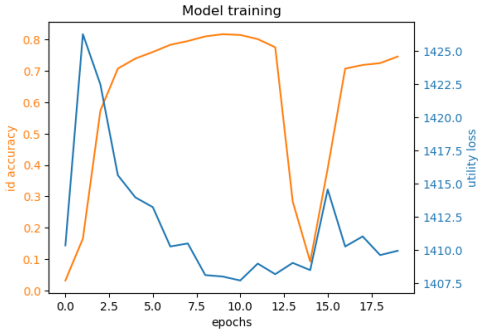
B.2 Results of the Adversarial Setting (*max_email_len*=200)



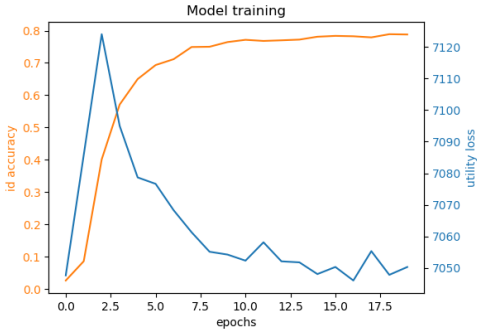
(a) ϵ=0.03



(b) ϵ=0.001

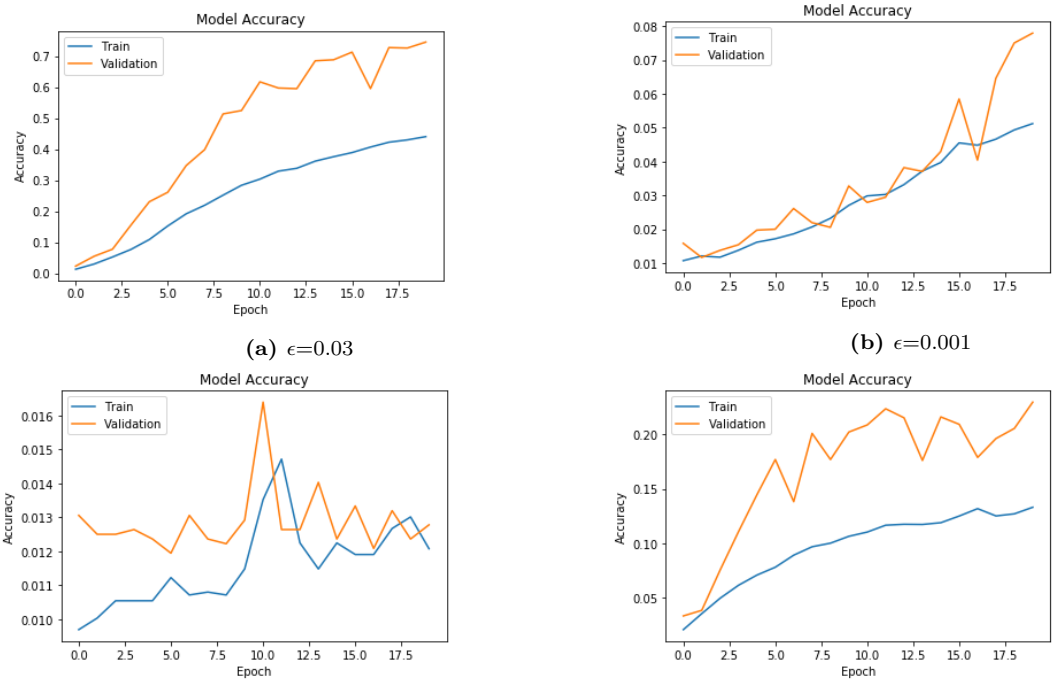


(c) ϵ=0.0001



(d) ϵ=0.00002

Figure 41: Learning Curve of the Anonymizer (*max_email_len*=200)



(a) ϵ=0.03 **(b)** ϵ=0.001

(c) ϵ=0.0001 **(d)** ϵ=0.00002

Figure 42: Learning Curve of the LSTM on Anonymized Data in the Adversarial Setting (*max_email_len*=200)

C Proof of Theorem 3.

Theorem 3. (Laplace mechanism for documents with Attention)
Given $\epsilon > 0$ and $n, m \in \mathbb{Z}^+$, let $\mathcal{M}_{\text{Att}} : \mathbb{R}^n \to \mathbb{DR}^{n\times m}$ be a mechanism that, given a matrix $X \in \mathbb{R}^{n\times m}$ and a matrix $A \in \mathbb{R}_+^{n\times m}$ outputs a noisy value as

$$X \mapsto X + A \cdot X',$$

where $X' \in \mathbb{R}^{n\times m}$ consists of i.i.d. random numbers distributed as $\text{Lap}(\frac{1}{\epsilon})$. The product of X' and A is conducted element-wise. Then $\mathcal{M}_{\text{Att}}$ satisfies $\dfrac{\epsilon}{\min\limits_{i,j}|a_{ij}|}||\cdot||$-privacy where $||\cdot||$ denotes the so-called Manhattan Metric computed as the sum of the element-wisely computed L_1-norm on $\mathbb{R}^n$.

Proof. To show the respective $\dfrac{\epsilon}{\min\limits_{i,j}|a_{ij}|}d_{\mathcal{X}}$-privacy guarantee the following must hold for any documents $X_1, X_2 \in \mathbb{R}^{n\times m}$ (especially those differing in one word only), any $\dfrac{\epsilon}{\min\limits_{i,j}|a_{ij}|}||\cdot|| \geq 0$ and a pseudometric $||\cdot||$:

$$\frac{\mathcal{M}_{\text{Att}}(S \subseteq \mathcal{X})(X_1)}{\mathcal{M}_{\text{Att}}(S \subseteq \mathcal{X})(X_2)} \leq \exp\left(\frac{\epsilon}{\min\limits_{i,j}|a_{ij}|}||\cdot|| \cdot ||X_1 - X_2||\right), \tag{66}$$

Let X_1, X_2 be two matrices containing word vectors as rows. After addition of a noise matrix containing $x'_{ij} \overset{\text{i.i.d.}}{\sim} \text{Lap}(\frac{1}{\epsilon})$ and element-wise multiplication with a matrix $A \in \mathbb{R}^{n\times m}$ as

$$x_{ij} \mapsto x_{ij} + a_{ij} \cdot x'_{ij}, \tag{67}$$

each entry of the anonymized document matrix is distributed as $x_{ij}^{\text{anonym}} \overset{\text{i.i.d.}}{\sim} \text{Lap}(x_{ij}\frac{a_{ij}}{\epsilon})$. The following derivation is conducted according to the earlier described proof of differential privacy for the Laplace Mechanism in [(1)]. For $z \in \mathbb{R}$ arbitrary,

$$\begin{aligned}\frac{\mathcal{M}_{\text{Att}}(x_{ij}^{(1)}, \epsilon)(z)}{\mathcal{M}_{\text{Att}}(x_{ij}^{(2)}, \epsilon)(z)} &= \frac{\frac{a_{ij}}{2\epsilon}\exp\left(-\frac{\epsilon|x_{ij}^{(1)}-z|}{a_{ij}}\right)}{\frac{a_{ij}}{2\epsilon}\exp\left(-\frac{\epsilon|x_{ij}^{(2)}-z|}{a_{ij}}\right)} = \exp\left(-\frac{\epsilon}{a_i j}(|x_{ij}^{(1)}-z| - |x_{ij}^{(2)}-z|)\right) \\ &= \exp\left(\frac{\epsilon}{a_{ij}}(|x_{ij}^{(2)}-z| - |x_{ij}^{(1)}-z|)\right) \overset{\text{triangle ineq.}}{\leq} \exp\left(-\frac{\epsilon}{a_{ij}}(|x_{ij}^{(1)}-x_{ij}^{(2)}|)\right).\end{aligned} \tag{68}$$

The result must somehow be transformed into an expression for the document X and not only one entry in the document matrix. As all entries are i.i.d., the joint distribution of all entries can be expressed as a product:

$$\begin{aligned}\mathcal{M}_{\text{Att}}(X, \epsilon)(z) = \prod_{i=1}^{n}\prod_{j=1}^{m}\mathcal{M}_{\text{Att}}(x_{ij}, \epsilon)(z) &= \prod_{i=1}^{n}\prod_{j=1}^{m}\frac{a_{ij}}{2\epsilon}\exp\left(-\frac{\epsilon|x_{ij}-z|}{a_{ij}}\right) \\ &= \exp\left(-\epsilon\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{|x_{ij}-z|}{a_{ij}}\right)\cdot\prod_{i=1}^{n}\prod_{j=1}^{m}\frac{a_{ij}}{2\epsilon}\end{aligned} \tag{69}$$

This computation is only feasible as A^{-1} does here not correspond to the inverse of A but to a matrix with all entries inverted: $A^{-1} = (\frac{1}{a_{ij}})_{ij}$. The above result is then extended to the document matrix as

$$\begin{aligned}\frac{\mathcal{M}_{\text{Att}}(X^{(1)}, \epsilon)(z)}{\mathcal{M}_{\text{Att}}(X^{(2)}, \epsilon)(z)} &= \frac{\exp\left(-\epsilon\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{|x_{ij}^{(1)}-z|}{a_{ij}}\right)\cdot\prod_{i=1}^{n}\prod_{j=1}^{m}\frac{a_{ij}}{2\epsilon}}{\exp\left(-\epsilon\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{|x_{ij}^{(2)}-z|}{a_{ij}}\right)\cdot\prod_{i=1}^{n}\prod_{j=1}^{m}\frac{a_{ij}}{2\epsilon}} \\ \leq \exp\left(\epsilon\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{|x_{ij}^{(1)}-x_{ij}^{(2)}|}{a_{ij}}\right) &\leq \exp\left(\epsilon\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{|x_{ij}^{(1)}-x_{ij}^{(2)}|}{\min_{i,j}|a_{ij}|}\right) \\ &= \exp\left(\frac{\epsilon}{\min_{i,j}|a_{ij}|}\sum_{i=1}^{n}\sum_{j=1}^{m}|x_{ij}^{(1)}-x_{ij}^{(2)}|\right)\end{aligned} \tag{70}$$

□

D Electronic Appendix

Preprocessing.R
kaggledata.R
corpusgen.R
StandardLength100.R
StandardLength200.R
Balance100.R
Balance200.R

Preprocessing_Tokenization.py
DataPrep_WordVec.py
Length100.py
Length200.py

AttentionLayer.py
Padding_Bias.py
NoiseLayer.py
AttnTensor.py

Models

Basic FFN

wo_att_dim60_len100.h5
wo_att_dim100_len200.h5

Identificator

att_dim50_len100.h5
att_dim60_len100.h5
att_dim80_len100.h5
att_dim100_len100.h5
att_dim50_len200.h5
att_dim60_len200.h5
att_dim80_len200.h5
att_dim100_len200.h5

LSTM

lstm_dim60_len100.h5
lstm_dim100_len200.h5

EvalManual

lstm_anon_100_0_03.h5
lstm_anon_100_0_001.h5
lstm_anon_100_0_0001.h5
lstm_anon_100_0_00002.h5

Anonymizer100

adv_model_dim60_100_emd2_0_03.h5
adv_model_dim60_100_emd2_0_001.h5
adv_model_dim60_100_emd2_0_0001.h5
adv_model_dim60_100_emd2_0_00002.h5

Anonymizer200

adv_model_dim100_200_emd2_0_03.h5
adv_model_dim100_200_emd2_0_001.h5
adv_model_dim100_200_emd2_0_0001.h5
adv_model_dim100_200_emd2_0_00002.h5

EvalAdversarial

lstm_adv_model_dim60_100_emd2_0_03.h5
lstm_adv_model_dim60_100_emd2_0_001.h5
lstm_adv_model_dim60_100_emd2_0_0001.h5
lstm_adv_model_dim60_100_emd2_0_00002.h5

Declaration of Authorship

I hereby declare that I have written the present thesis independently and have not used any tools other than those indicated. The places of the paper, which were taken from other sources in the wording or the sense, are marked by data of the origin. This also applies to drawings, sketches, pictorial representations and sources from the Internet.

Location Date

Signature (Lina Fischer)