An Approach Towards Unsupervised Text Simplification on Paragraph-Level for German Texts

Master Thesis

in the Course of Studies Applied Computer Science
in the Faculty Information Systems and Applied Computer Sciences
of the Otto-Friedrich-University Bamberg

Chair of Media Informatics

Submitted by: Leon Fruth

Supervisor: Prof. Dr. Andreas Henrich

Tutor: Robin Jegan, M.Sc.

Bamberg 2026

URN: urn:nbn:de:bvb:473-irb-112994x
DOI: https://doi.org/10.20378/irb-112994

Abstract

Text simplification aims to make texts easier to read and comprehend for people. Recent approaches tackle this task by training neural models on large-scale parallel datasets. However, most languages only have limited simplification data available. Laban et al. [2021b] presented an unsupervised simplification approach to avoid the need for parallel data. They introduced the training algorithm k-SCST, which optimizes a reward by generating and scoring multiple candidate simplifications and encouraging the candidates that outperform the mean reward. This thesis adapts this approach to simplify short paragraphs from German Wikipedia articles. The individual scores of the reward regarding simplicity, fluency and meaning preservation are modified for the new domain and language. In addition, other aspects of the training method are explored. The results show some lexical and syntactic simplification phenomena but also problems regarding fluency and faithfulness. The findings are assessed, and suggestions for future improvements are presented.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Procedure and Structure . . . 2

2 Background **3**
- 2.1 Introduction to Automatic Text Simplification . . . 3
  - 2.1.1 Lexical Simplification . . . 4
  - 2.1.2 Syntactic Simplification . . . 4
  - 2.1.3 Simplification Measurements . . . 5
  - 2.1.4 Simplification as Machine Translation task . . . 7
  - 2.1.5 Unsupervised Simplification . . . 9
  - 2.1.6 German Text Simplification . . . 10
- 2.2 Reinforcement Learning . . . 11
  - 2.2.1 Reinforcement Learning in NLP . . . 12
  - 2.2.2 Self-critical Sequence Training . . . 12

3 GUTS **14**
- 3.1 Reward . . . 15
  - 3.1.1 Simplicity . . . 17
  - 3.1.2 Meaning Preservation . . . 21
  - 3.1.3 Fluency . . . 24
  - 3.1.4 Guardrails . . . 27
- 3.2 Generator . . . 30
  - 3.2.1 Learning . . . 31
  - 3.2.2 Candidate Sampling . . . 32

4 Experiments **37**
- 4.1 Data . . . 37
- 4.2 Setup . . . 39
- 4.3 Training . . . 41

5 Evaluation **45**
- 5.1 Automatic Evaluation . . . 45
  - 5.1.1 Data . . . 45
  - 5.1.2 Metrics . . . 47
  - 5.1.3 Models . . . 47
  - 5.1.4 Results . . . 48

5.2 Manual Evaluation 49
5.2.1 Simplification phenomena 51
5.2.2 Problems 54

6 Discussion **57**
6.1 Contributions 57
6.2 Limitations 58
6.3 Future Work 60

7 Conclusion **62**

Bibliography **62**

A Simplification Examples **70**
A.1 Pivot Model 70
A.2 Model Simplifications 71

List of Tables

1 FRE scores and the corresponding U.S. grade level classifications 6

2 Example article from the GWW dataset 16
3 Examples of factual inconsistency from generated by GUTS 30

4 Training data used for the experiments 37
5 Score weights used for the training runs 40

6 Examples from the TextComplexityDE evaluation data. 46
7 Automatic results of TextComplexityDE 49
8 Automatic results of Wikipedia paragraphs 49

9 Overview of transformer models used for GUTS 59

List of Figures

1 The transformer architecture [Vaswani et al., 2017] 8

2 The GUTS Training Framework 14
3 The GUTS reward composition 15
4 Plot of mean Zipf values for removed and added words 17
5 Example for the calculation of $S_{lexical}$ 19
6 Flesch Reading Ease values of the reference datasets 20
7 Example for the calculation of $S_{syntactic}$ 21
8 Sentence alignment example 22
9 Illustration of the BERTScore computation with a candidate and reference sentence [Zhang et al., 2019] 23
10 Incoherent simplification reaching a fluency score of **0.96** 26
11 Hallucination detection algorithm 29
12 The GUTS Learning with k-SCST 31
13 Peaked and flat distributions with Top-K sampling ($K = 5$) 34
14 Peaked and flat distributions with nucleus sampling ($n = 0.95$) 35

15 Difficult example paragraphs from Wikipedia 38
16 Performances of k-SCST with different values for k [Laban et al., 2021b] . . 39
17 First 24 hours of training progression 41
18 Reward optimization with different learning rates 42
19 Plot of $S_{article}$ after introducing it to GUTS-1 42
20 Plot of $S_{hallucination}$ for GUTS-1 and GUTS-2 43
21 Example samples with rewards from training GUTS-2 44

22 Simplification example of the Pivot model 48
23 Simplifications examples from the evaluated models 50
24 Excerpts of lexical substitutions by GUTS-2 51
25 Excerpts of word reductions by GUTS-2 52
26 Sentence splittings generated by GUTS-2 53
27 Deletions performed by GUTS-2 53
28 Grammatical errors produced by GUTS-2 54
29 Date and numeric mistakes produced by GUTS-2 55
30 Hallucinations generated by GUTS-2 55
31 Faithfulness error generated by the Pivot model 56

32 Pivot model example 70
33 Simplification examples 1 71

34 Simplification examples 2 72
35 Simplification examples 3 72

List of Abbreviations

ATS	Automatic Text Simplification
SCST	Self-Critical Sequence Training
RL	Reinforcement Learning
GUTS	German Unsupervised Text Simplification
NLP	Natural Language Processing
SARI	System output against references and against input sentence
FKGL	Flesch-Kincaid Grade Level
SAMSA	Simplification automatic evaluation measure through semantic annotation
FRE	Flesch Reading Ease
GFI	Gunning-Fog-Index
MT	Machine Translation
BLEU	Bilingual evaluation understudy
NIST	National Institute of Standards and Technology
RNN	Recurrent neural network
LSTM	Long short-term memory
BERT	Bidirectional Encoder Representation from Transformers
ACCESS	AudienCe-CEntric Sentence Simplification)
DRESS	Deep Reinforcement Sentence Simplification
KiS	Keep it Simple (refers to the work from Laban et al. [2021b])
GPT-2	Generative Pre-trained Transformer 2
MDP	Markov Decision Process
ROGUE	Recall-Oriented Understudy for Gisting Evaluation
GWW	Gemeinnützige Werkstätten und Wohnstätten
AMP	Automatic Mixed Precision
GAN	Generative Adversarial Network
NER	Named Entity Recognition
GPU	Graphic processing unit
LM	Language model

Notation

General

P	original paragraph written by a human
S	simplification performed by a system
$S_{lexical}$	score measuring the lexical simplicity
$S_{syntactic}$	score measuring the syntactic simplicity
$S_{meaning}$	score rating the meaning preservation of S
$S_{fluency}$	score rating the fluency by a LM
S_{discr}	score predicting how human-like S is with a Discriminator
$S_{brevity}$	guardrail score for the text length
$S_{hallucination}$	guardrail score penalizing the introduction of new entities
S_{ngram}	guardrail score penalizing the repetition of n-grams
$S_{articles}$	guardrail score penalizing repetition of different articles
t	time-step
R^{Sj}	reward of a candidate sequence index j
$\bar{R}^{S}$	mean reward of multiple candidate sequences
L	training loss
w^{Sj}	sampled candidate sequence j
w_i^{Sj}	i-th word from sampled sequence
$w_{<i}^{Sj}$	preceding word(s) of the i-th word
$p(w_i^{Sj}\|w_{<i}^{Sj}, P)$	probability of w_i^{Sj} conditioned on $w_{<i}^{Sj}$ and P
K	value for top-K sampling
p	value for nucleus sampling
$\mathcal{V}$	vocabulary

Chapter 2

S^R	simplification reference
N_{words}	number of words in a text
$N_{sentences}$	number of sentences in a text
$N_{syllables}$	average number of syllables per word
D	number of words with three or more syllables in a text
$\hat{R}$	reward obtained by a model
$\hat{w}$	word sequence sampled by greedy decoding
$w^S = (w_1^S, ..., w_T^S)$	sampled sequence of T words
$w_{<i}$	preceding word(s) of the i-th word
$p(w_i^S\|w_{<i}, P)$	probability of the i-th word conditioned on $w_{<i}$ and P
R^S	reward for a sampled sequence

Chapter 3

R	reward of a simplification
s_i	individual score of the reward
W_i	weight of a score
$\overline{Zipf}_{add}$	average Zipf value of the set containing the added words
$\overline{Zipf}_{rem}$	average Zipf value of the set containing the removed words
$sent_i^P$	individual sentence from an original paragraph
$sent_i^S$	individual sentence from a simplification
$sents^P$	list of sentences from an original paragraph
$sents^S$	list of sentences from a simplification
$x = x_1, x_2, ..., x_n$	reference sentence with n tokens
$\hat{x} = \hat{x}_1, \hat{x}_2, ..., \hat{x}_n$	candidate sentence with n tokens
R_{BERT}	recall of BERTScore
P_{BERT}	precision of BERTScore
F_{BERT}	F1-Score of BERTScore
$x_i^\top \hat{x}_j$	cosine similarity between x_i and $\hat{x}_j$
$aligned$	list of aligned sentence tuples
$unaligned$	list of unaligned sentences
w	sequence of words $(w_1, ..., w_t)$
$LM(w)$	function for returning the language model probability for a word sequence
$w_{\setminus t}$	all words of a sequence but the t-th word
Θ	parameters of a language model
$\log p(w_t \mid w_{\setminus t}; \Theta)$	log probability for a word w_t conditioned on $w_{\setminus t}$ and Θ
λ	parameter for tuning $S_{fluency}$
$p_{discr}(S)$	a discriminator model's probability for S being authentic
$L(\cdot)$	function returning the length of a text
C	compression rate of S compared to P
C_{min}, C_{max}	minimum and maximum compression rate for $S_{brevity}$
w	sequence of words $(w_1, ..., w_t)$
$p(w_t \mid w_{<t})$	probability distribution of a word w_t, conditioned on the previous words
q_i	probability of the i-th word from the vocabulary
z_i	logit vector of the i-th word from the vocabulary
$temp$	temperature value
$exp(\cdot)$	exponential function of a value
$\mathcal{V}^{(K)}$	set of the K most likely words from a vocabulary
$\mathcal{V}^{(n)}$	set of the words from V with a cumulative probability of n
$H(p(\cdot \mid w< t))$	conditional entropy of a models probability distribution
τ	absolute probability range value for typical decoding

Chapter 5

$FRE(\cdot)$	function returning the Flesch Reading Ease of a text
$FRE\ diff$	difference between $FRE(S)$ and $FRE(P)$
$\overline{Zipf}(\cdot)$	average Zipf value of all filtered words in a text
$Zipf\ diff$	difference between $\overline{Zipf}(S)$ and $\overline{Zipf}(P)$
$Comp.$	compression rate of a simplification

Chapter 1

Introduction

1.1 Motivation

The objective of text simplification is the modification of texts in a way to make them simpler to read and understand for the target audience. Automatic text simplification (ATS) is a research field in computational linguistics. Work towards this field studies methods and techniques to simplify texts. It is closely related to other natural language processing (NLP) tasks such as text summarisation. The objective of text summarisation is to reduce a text to its essential content, while text simplification aims to preserve most of the relevant information. [Saggion, 2017]

A main motivation for ATS is to make information available to a wider audience by helping people with low literacy levels, mentally impaired people and children [Al-Thanyyan and Azmi, 2021]. Research has shown the positive impact of text simplification for people with dyslexia [Rello et al., 2013], autism [Evans et al., 2014], aphasia [Carroll et al., 1999] or people facing reading difficulties, such as children and non-native speakers [Watanabe et al., 2009]. Especially long and syntactically complex sentences, as well as the usage of complicated and infrequent words can be hard to process, which has been proven to be an obstacle for people with aphasia and autism. [Carroll et al., 1999; Evans et al., 2014] In the information age, various texts play an important role in the everyday lives of end-users. Simplifications can improve the quality of life for users and authors since the user's understanding of texts improves, and the author's written work can reach more people. [Shardlow, 2014]

Text simplification can be broken down into lexical and syntactic simplification. Lexical simplification attempts to modify the vocabulary of texts, to be more appropriate for the reader. Syntactic simplification aims to increase the readability of texts by creating shorter sentences or avoiding complicated constructions. Those tasks are often addressed separately but are naturally related.

With the advancements in deep learning, newer work approaches ATS as a monolingual machine translation problem: Translating a text with complex linguistic properties into a text with simple linguistic properties in the same language. Instead of following simplification rules, statistical or neural models rely on representation learning to capture simplification phenomena from data. They are generally trained with parallel examples of complex-simple sentence pairs. Training this way requires large amounts of such parallel simplification data. [Mallinson et al., 2020] Large-scale simplifications datasets exist for

some languages, like English or Spanish [Xu et al., 2015; Agrawal and Carpuat, 2019]. Other languages only have a limited amount of simplification data available, which is often insufficient to train a model.

Laban et al. [2021b] presented an approach to bypass the need for parallel datasets by using reinforcement learning (RL)-based training. Their work shows a reference-free reward regarding the criteria simplicity, meaning preservation/salience and fluency, that are jointly optimised. For training, they introduce a novel RL algorithm, named *k-SCST*, an extension of Self-Critical Sequence Training (SCST) [Rennie et al., 2017]. With this procedure a model learns to simplify whole paragraphs of English text instead of sentence-level simplification.

This approach is taken and adapted to the German language in the following work. Since German simplification data is limited, the dependency on a large parallel simplification dataset is circumvented. This work presents the first unsupervised ATS approach for German to the author's knowledge. Moreover, almost none German simplification approaches exist that simplify on a paragraph-level.

1.2 Procedure and Structure

In Chapter 2, the theoretical background for this work is described. First, some past and recent approaches for ATS are outlined. Next, a quick overview of reinforcement learning is presented. Here some applications for natural language processing tasks besides text simplification are described, then the training algorithm used in this work is presented.

Chapter 3 will present the German ATS system GUTS, short for **G**erman **U**nsupervised **T**ext **S**implification. Because GUTS is based on the work from Laban et al. [2021b], this chapter will dissect each component of their work. Furthermore, it will be explained how their approach is adapted to the German language for this thesis. First, the individual scores for the reward are described. Secondly, the learning process of the generator, which is responsible for producing the simplifications, is outlined.

The subsequent Chapter 4 presents the experiments conducted in this work. This chapter will present the data and setup used during the experiments and training runs. Moreover, some observations and problems that arose during these runs are highlighted.

Two of these trained GUTS models are then evaluated in Chapter 5. Firstly, the models are evaluated and compared using automatic metrics. Then a manual evaluation to highlight observed simplification phenomena and problems with the produced simplifications of GUTS are conducted. The evaluated models have shown problems concerning the factuality and fluency of the simplifications. Despite slow training time, some simplification phenomena could be observed with the generated simplifications.

The findings are then discussed in Chapter 6, where the results are reflected concerning the contributions, limitations and possible improvements for future work. Overall, the work presents an approach that can be improved regarding various aspects and adapted to different domains and languages.

The last chapter will conclude this thesis by shortly recapitulating what has been done and what results have been found.

Chapter 2

Background

In the following chapter, the theoretical background for this work is outlined. First, automatic text simplification (ATS) is introduced. Here different approaches are described, from rule-based simplification solutions to neural models that tackle the problem as a machine translation task, given sufficient data to train on. Next, some approaches are looked at that avoid the need for large parallel datasets by training in an unsupervised way. Lastly, an overview of German research regarding text simplification is presented.

Next, a quick introduction for reinforcement learning (RL) is presented, with a focus on advances in the field of natural language processing (NLP). At last, the reinforcement learning framework k-SCST is examined, an ablation of self-critical sequence training that is used in this work.

2.1 Introduction to Automatic Text Simplification

ATS describes the process of modifying a text in natural language to reduce its linguistic complexity and increase its understandability and readability. [Shardlow, 2014]

Since the late '90s, ATS has been approached from different angles and has followed the growth of machine learning and natural language processing. It is an active research area with continuous improvements, which shows that text simplification is still far from a satisfactory solution. [Al-Thanyyan and Azmi, 2021] Within the field of NLP, ATS has similarities to other techniques like machine translation, text-to-text generation, paraphrase generation and text summarisation. These fields share techniques and resources from each other. [Shardlow, 2014] Primarily text summarization is closely related to ATS and can use similar methods. The approach adopted for this work from Laban et al. [2021b] was first released in the context of summarization [Laban et al., 2021a] and then adapted to text simplification. Summarization focuses on reducing the length of a text and only capturing the most important information of the text. For simplification, the texts are often shorter than the original [Petersen and Ostendorf, 2007], but this is not always true. An objective in simplification is to preserve as much content from the original text as possible [Al-Thanyyan and Azmi, 2021].

Following [Laban et al., 2021b] in this thesis a text simplification should balance the following properties:

1. **Simplicity:** The simplification of a text should be syntactically and lexically simpler than the original.

2. **Fluency:** The simplified text should be a well-formed German simplification.

3. **Meaning Preservation:** The simplification should contain the same information as the original text. Also referred to as salience, meaning adequacy or relevance in other work.

Regarding simplicity in ATS, the two main tasks are lexical and syntactical simplification. Both of these are naturally related. If the syntactic structure of a sentence is changed, it might be necessary to perform transformations on a lexical level to keep the sentence grammatical. Vice versa, a lexical simplification might require syntactic restructuring. Any change on a local level affects other parts of a text. For example, this might occur when replacing a masculine noun with a feminine synonym, which requires some changes to pronouns, adjectives, or even to preceding and following sentences. [Saggion, 2017]

2.1.1 Lexical Simplification

The lexical simplification of written text can be achieved by replacing difficult words with simpler expressions that are semantically equivalent. For example, the German word "verzehren" (English: "consume") can be lexical simplified with "essen" (English: "eat") without changing the original meaning of a text. Moreover, the removal of superfluous words or the explanations of complex words can contribute to lexical simplification. [Keskisärkkä, 2012]

To identify complex words and choose simpler words or phrases as substitutions, one must decide how to measure word complexity. Some work has been measuring complexity by relying on the relative word frequencies [Carroll et al., 1998; Laban et al., 2021b; Keskisärkkä, 2012]. Word frequency has been shown to correlate strongly with word difficulty but is not always a guarantee [Breland, 1996]. Keskisärkkä [2012] has shown that word length can be used as a measurement for word difficulty.

The first rule-based approach for lexical simplification was proposed by Carroll et al. [1998] to simplify English news articles for aphasic readers. The system consists of a linguistic analyser for the articles and a lexical and syntactic simplifier. After the text was analysed, the lexical simplifier used the most frequently occurring synonym received from a database for simplification.

Qiang et al. [2021] presented a context-aware lexical simplification system that uses the contextual word embeddings produced by their Bidirectional Encoder Representation from Transformer (BERT) model to generate and rank simplification candidates in combination with word frequency and a paraphrase database.

This work uses the implementation from Laban et al. [2021b], which uses the correlation between word frequency and complexity as a basis, to construct a score to measure the lexical simplicity.

2.1.2 Syntactic Simplification

The goal of syntactic simplification is to identify syntactic phenomena in sentences, which are hard to read and comprehend for some readers and rewrite them with simpler and more understandable language. [Saggion, 2017] For example, long sentences, frequent use

of passive or long sequences of adjectives can be difficult for aphasic readers. [Carroll et al., 1999]

A common example for a syntactic simplification would be sentence splitting. E.g. the sentence "Die Katze Momo jagd eine Maus und legt sie vor die Haustüre." can be split into "Die Katze Momo jagd eine Maus. Momo legt sie vor die Haustüre.". Each sentence now carries only one statement. Next to sentence splitting, syntactic operations like sentence reordering, information addition, sentence joining, and content selection can help readability. Some of these operations need to be performed on a paragraph- or document-level and can not be achieved by simplifying sentences individually. [Alva-Manchego et al., 2019]

An early system for syntactic simplification was introduced by Chandrasekar et al. [1996]. First, they retrieve a structural representation of the sentence. Then a sequence of rules is applied to identify and extract text parts that can be simplified. This approach provided a foundation for more rule-based simplification approaches [Saggion, 2017].

Again, following the work of Laban et al. [2021b], a score for the syntactic simplicity $S_{syntactic}$ as part of the reward is constructed. This score is based on the metric Flesch Reading Ease, presented in the following section.

2.1.3 Simplification Measurements

Different measurements exist to evaluate various aspects of text simplification, yet there is no single agreed-upon metric. Some of these are used for automatic evaluation, which is commonly done by using SARI or Flesch-Kincaid Grade Level (FKGL).

SARI, short for **S**ystem output **A**gainst **R**eferences and against **I**nput sentence, is a reference-based automatic metric designed for tuning and evaluating simplification systems. It compares the lexical simplification produced by a system with the original text and human-written simplification references. With SARI, the following operations are considered [Xu et al., 2016]:

1. *Addition* operations are rewarded, where words from the system's simplification S were not in the original paragraph P but occurred in the simplification reference S^R.
2. *Retained* words that occur in both P, S, and S^R positively impact the SARI.
3. *Deleted* words that are in P, but were removed in both S and S^R improve the metric.

SARI is correlated with simplicity gains, when the reference is produced through lexical paraphrasing [Alva-Manchego et al., 2021].

The problem with reference-based simplification methods is that every sentence has a large space of possible valid simplifications. Simplification datasets so far only contain one or a few example simplifications. Reference-less measurements like **S**implification **A**utomatic evaluation **M**easure through **S**emantic **A**nnotation (*SAMSA*) avoid this problem. An optimal structure for SAMSA is where each predicate-argument structure is assigned to a single sentence. It measures whether the input sentence was split towards this optimal granularity. Additionally, it measures how well the meaning of the source sentence is preserved in the output simplification. [Sulem et al., 2018b] SAMSA primarily focuses on sentence splitting operations, Alva-Manchego et al. [2021] showed in their experiments that SAMSA had a low correlation with structural simplicity. They argue that this metric should only be used in examples where sentence splitting was performed.

Score	School level (US)
100-90	5th grade
90-80	6th grade
80-70	7th grade
70-60	8th & 9th grade
60-50	10th to 12th grade
50-30	College
30-10	College graduate
10-0	Professional

Table 1: FRE scores and the corresponding U.S. grade level classifications

Flesch-Kincaid grade level (FKGL) was developed to measure the readability of texts and offers a simple reference-less method to rate syntactic simplicity. To calculate the score, the average number of syllables per word and average words per sentence are considered:

$$FKGL = 0.39\frac{N_{words}}{N_{sentences}} + 11.8\frac{N_{syllables}}{N_{words}} - 15.59 \tag{1}$$

With this, shorter sentences with shorter words (fewer syllables) achieve a higher FKGL score. The result is represented in U.S. grade levels. [Flesch, 1943]

An ablation of this is the *Flesch Reading Ease* (FRE) which redistributes the scores between 0 and 100 (Table 1). [Flesch, 1948]

$$FRE = 206.835 - 1.015(\frac{N_{words}}{N_{sentences}}) - 84.6(\frac{N_{syllables}}{N_{words}}) \tag{2}$$

Amstad [1978] later transferred the FRE from English to German:

$$FRE_{german} = 180 - \frac{N_{words}}{N_{sentences}} - 58.5(\frac{N_{syllables}}{N_{words}}) \tag{3}$$

Tanprasert and Kauchak [2021] argues that FKGL is unsuitable for the evaluation of a system's output for simplification, as it was designed to measure human output. Because of the simplistic nature of this metric, it is very easy to cheat. [Tanprasert and Kauchak, 2021]

This work constructs a score with the FRE_{german} for the reward that tries to capture the syntactic readability of the simplifications produced by the presented system. In the course of this work, the FRE_{german} will be referred to as FRE.

Another metric similar to FKGL is the *Gunning-Fog-Index* (GFI) introduced by Gunning et al. [1952]. It also returns the school grade for which the text was written or should be targeted.

$$GFI = (\frac{N_{words}}{N_{sentences}} + 100\frac{D}{N_{words}}) * 0.4 \tag{4}$$

Where D denotes to the number of words in the text with three or more syllables. The GFI functions similarly to the FKGL or FRE, but unfortunately, no German adaption is available for the Gunning-Fog-Index.

2.1.4 Simplification as Machine Translation task

ATS can also be addressed through the lens of machine translation (MT), where a complex text of a language is translated into a text of the same language with simpler linguistic properties. [Al-Thanyyan and Azmi, 2021]

With the introduction of large-scale parallel corpora such as WikiLarge [Xu et al., 2015] and Newsela[1], several monolingual MT approaches for ATS have been proposed. WikiLarge is a collection of sentences from English Wikipedia articles aligned with the corresponding sentence from Simple Wikipedia articles [Xu et al., 2015]. Newsela is a collection of news articles and multiple simplifications for different school grade levels. While there also exist some larger datasets for Spanish simplification [Agrawal and Carpuat, 2019], other languages only have very limited parallel datasets that are mostly insufficient to train a simplification model in a MT fashion.

The first attempt to treat simplification as a translation task by Specia [2010] used statistical MT to learn to simplify Portuguese. They used Moses [Koehn et al., 2007], a phrase-based statistical MT system. They trained the system by translating complex sentences to the aligned simplified sentences with two different simplification levels. While the results were limited by the small training corpus with around 4,000 sentence pairs, they showed that their system had already learned to use some simplification procedures, mostly of lexical nature. The generated and human-written simplifications were evaluated on MT evaluation metrics BLEU [Papineni et al., 2002] and NIST [Doddington, 2002]. Also, a human evaluation was conducted, which showed promising results for the criteria fluency, adequacy (meaning preservation) and simplicity. [Specia, 2010]

With good results of deep learning techniques in MT and other NLP tasks, recurrent neural network (RNN) [Rumelhart et al., 1985] and long short-term memory (LSTM) networks [Hochreiter and Schmidhuber, 1997] have been employed for learning text simplification. A LSTM encoder-decoder structure was investigated for text simplification by Wang et al. [2016]. The authors found that their LSTM model can learn sentence- and word-level operations, like sorting, reversing and replacing.

With the introduction of *transformer* networks, most NLP tasks' state-of-the-art improved. The original transformer model, proposed by Vaswani et al. [2017] for translation, uses an encoder-decoder architecture. The architecture is visualized in Figure 1, the left box being the encoder, the right part the decoder. The encoder represents the input as a contextual embedding, and the decoder generates the output sequence based on this embedding. Positional encodings first label the input by appending a number to each word signalling the order of the words. This way, they do not need to rely on recurrence and memory, like RNNs or LSTMs, to capture the sequential nature of the input. Instead, transformers rely solely on their attention mechanism to draw global dependencies between the inputs and output. The encoder and decoder consist of a stack of N identical layers. Each encoder layer has a multi-head attention mechanism and a fully connected feed-forward network followed by layer normalization. The decoder has the same components as the encoder but an additional masked multi-head attention layer. These properties allow parallelization of the input data for transformers. Instead of receiving the input sequentially, like recurrent model architectures, transformers receive the whole input sequence at once. This enables a significant speed increase compared to recurrent networks, especially when handling longer sequences. Training is thereby faster

[1] https://newsela.com/

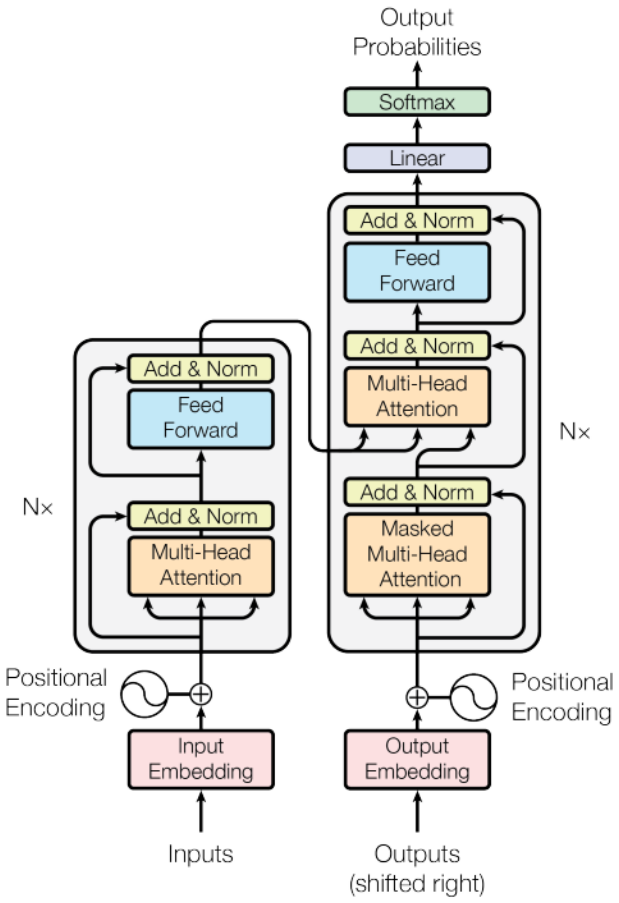


Figure 1: The transformer architecture [Vaswani et al., 2017]

and allows the processing of larger datasets. [Vaswani et al., 2017]

This new architecture led to the creation of large pre-trained models such as BERT, short for Bidirectional Encoder Representation from Transformers. BERT is a model entirely consisting of stacked transformer encoder blocks, which is pre-trained in a self-supervised way by trying to predict randomly masked words. BERT can be further fine-tuned on several tasks, such as sentiment analysis, textual similarity or question answering. [Devlin et al., 2018]

Martin et al. [2019] introduced ACCESS, a sentence simplification approach based on the original encoder-decoder transformers architecture from [Vaswani et al., 2017]. They introduced a parametrization mechanism to control the compression rate, paraphrase amount, and the strength of lexical and syntactic simplification. By controlling these attributes, the system can potentially be adapted to the needs of different audiences. Their system was trained on the large parallel simplification dataset WikiLarge [Xu et al., 2015] and achieved state-of-the-art SARI.

2.1.5 Unsupervised Simplification

Several methods tackle text simplification in an unsupervised way. With data availability being one of the main problems in ATS, unsupervised approaches bypass the need for large-scale parallel data. Moreover, simplification datasets often offer only one or few possible simplifications for a sentence. These only cover a small part of a sentence's large space of valid simplifications [Sulem et al., 2018b].

Zhang and Lapata [2017] introduce the model DRESS, which is based on RNN encoder-decoder architecture. The encoder transforms a source sentence into a continuous-space representation, which the decoder uses to generate a simplification. The model is trained using the reinforcement learning framework REINFORCE [Williams, 1992] with a reward function that encourages simplicity, fluency, and relevance. Simplicity was measured by the metrics SARI, which is outlined in Section 2.1.3. Fluency measures the grammaticality and how well-formed the simplified text is. They represent the fluency by computing the normalized sentence probability for a generated simplification with an LSTM that was trained on simple sentences. The relevance is measured by another LSTM sentence encoder that converts source and simplified sentences into vectors. The cosine similarity between those vectors makes up the relevance score for the reward. The cosine similarity is a measure of similarity between two vectors A and B defined as:

$$\frac{A \cdot B}{\parallel A \parallel \parallel B \parallel} \tag{5}$$

where the dot product $\cdot$ is divided by the magnitude of the vectors.

Nakamachi et al. [2020] use a similar approach with an LSTM encoder-decoder model to simplify sentences. Generated simplifications are scored with a reward function and an adaption of the REINFORCE algorithm is used for optimizing the reward. The reward takes into account simplicity, meaning preservation and grammaticality. However, instead of using evaluation metrics for text simplification like BLEU, SARI and FKGL, they fine-tune three pre-trained BERT models [Devlin et al., 2018] on different datasets. For grammatically, one BERT model is trained on a dataset to estimate a sentence's grammaticality. Meaning preservation is estimated with a BERT model trained on STS-B [Cer et al., 2017], a dataset for evaluating the semantic similarity of sentence pairs. For simplicity, they use the third BERT model to estimate the reading level of simplification sentences from the Newsela dataset. Their evaluation showed that humans preferred the results of their system over other models such as the previously explained DRESS [Zhang and Lapata, 2017] model.

Keep it Simple

The approach used in this work is adapted from the work of Laban et al. [2021b] and will be referred to as *KiS* in this thesis, short for **K**eep **i**t **S**imple. Instead of simplifying individual sentences, they train with whole paragraphs of texts since it has been shown that common simplification phenomena happen on a paragraph-level [Alva-Manchego et al., 2019]. As a generator model, which performs the simplifications, they use the model **G**enerative **P**re-Trained **T**ransformer 2 (GPT-2) from Radford et al. [2019] to produce simplifications from a source text.

GPT-2 is a large transformer-based language model. In KiS, they use the medium version of GPT-2 with 345M Parameters, 24 transformer decoder blocks stacked on top of each

other and a model dimensionality of 1024. The model is pre-trained in a self-supervised way by predicting the next word given the previous words of a text. With training on large collections of texts, GPT-2 is able to achieve state-of-the-art performance on a variety of domain-specific language modelling tasks. Even without fine-tuning, the model achieves promising results on tasks such as question answering, translation or summarization and other text generation tasks. GPT-2 can be further fine-tuned for these tasks to achieve even better results. These capabilities make it a good candidate for text simplification, the task at hand. [Radford et al., 2019]

Each simplification in KiS is rewarded using different scores for syntactic and lexical simplicity, fluency, salience or meaning preservation and some guardrail scores, which try to mitigate common text generator problems such as hallucination facts or the repetition of words. The rewards are further described in Chapter 3, and how they are adapted for this work. For training they use an ablation of **s**elf-**c**ritical **s**equence **t**raining (SCST) to optimize the rewards. SCST, a form of the REINFORCE algorithm, is further explained in Section 2.2.2. The model is trained on 7 million English news articles. They compare their results against other simplification models on the Newsela corpus and further construct a novel human comprehension study to evaluate simplicity. The KiS model achieves state-of-the-art results on automatic evaluation and the best results in their conducted study.

2.1.6 German Text Simplification

ATS research for the German language is pretty limited. Some small parallel datasets exist, such as *TextComplexityDE*, which has 250 complex-simple sentence pairs and is used for as reference in this work. [Naderi et al., 2019]

There exist further datasets that are unreleased or not publicly available at the time of writing this thesis. For example Klaper et al. [2013] created a simplification dataset by scraping parallel articles from the web. The articles are received from websites that offer the texts in multiple versions, referred to as *Alltagssprache* (English: "everyday language") and *Leichte Sprache* (English: "simple language"). They align the simple-complex pairs and collect around 7000 parallel sentences from these resources.

This corpus was enhanced by Battisti and Ebling [2019], where the parallel data was increased to 17,121 simple-complex sentence pairs. Moreover, they added 5,461 monolingual documents in simplified German to the corpus, containing additional information on text structure, typography and images.

The recently released work from Rios et al. [2021] presented a parallel dataset containing news articles from the Swiss news magazine *20 Minuten*. This dataset consists of full articles paired with simplified summaries.

Säuberli et al. [2020] introduced a new parallel dataset for simplification, collecting 3616 sentence pairs from the Austria Press Agency. They trained multiple transformer-based models on that dataset. They showed that their dataset is not large enough to sufficiently train a neural machine translation system that produces adequate and fluent simplifications. Still, the model showed some simplification features, even with such limited data.

Suter et al. [2016] presented a rule-based simplification approach that is not dependent on simplification data. The rules are compiled according to German simplification guidelines [Maaß, 2015]. The rules are divided into character- and word-level, sentence-level, text-level and layout rules. Compared to many other ATS approaches, it performs

the simplification at a corpus-level instead of simplifying individual sentences. Through qualitative and quantitative evaluation, they showed that their system has problems with reducing the lexical simplicity through substitutions. However, with added explanations they arguably achieved an increased lexical simplicity. The syntactic transformation performed by their system were comparable to that of human simplifications.

Mallinson et al. [2020] introduce a zero-shot cross-lingual simplification approach. They utilize the availability of large-scale English sentence-simplification datasets to sidestep the insufficient size of German data. They train a multilingual transformers model on (1) a simplification dataset, (2) translation task (English-German) and (3) a language modelling task to take advantage of available data. This method transfers simplification knowledge from a high-resource language (English) to a low-resource language (German). The model was able to outperform other models on German simplification tasks, shown in their automatic and human evaluation. This approach showed the cross-lingual capabilities of modern transformer models and the possibility of learning German simplifications by primarily relying on English simplification data. Furthermore, the model can be used to combine the translation task between English and German with the simplification task.

This work implements another way to side-step the data scarcity for German ATS, by using an unsupervised approach with reinforcement learning. To the author's knowledge, this is the first approach that trains a neural model in an unsupervised way for text simplification. Additionally, it is one of the first times a neural model is used to simplify German texts on a paragraph-level instead of a sentence-level.

2.2 Reinforcement Learning

Reinforcement learning (RL) is a branch of machine learning that is inspired by the learning process of humans and animals. RL is used to teach an agent a goal-oriented task without providing examples of the right action. Instead, it receives a reward to indicate whether it performed good or bad in this situation. [Ng, 2003]

RL problems are often described as a Markov Decision Process (MDP). Each time-step t the agent is in a state s_t from a state space $\mathcal{S}$ and selects an action a_t from the action space $\mathcal{A}$. The behaviour of the agent can be described as a policy $\pi(a_t|s_t)$, mapping an action a_t to a state s_t. For every performed action the agent receives an reward r_t computed by the reward function $\mathcal{R}(s_t, a_t)$ and transitions to the next state s_{t+1}. [Li, 2017]

Famous examples in the history of RL are often applied to games like chess or Go. With Deep Blue, IBM created a chess-playing model that defeated the World Chess Champion Garry Kasparov [Campbell et al., 2002]. Another well-known example is AlphaGo Zero by DeepMind, a software to play the highly complex game Go, which it mastered by playing only against itself. It outperformed its predecessor, which was trained partially with RL and partially with supervised learning using human expert data. [Silver et al., 2017]

With the incorporation of deep learning into RL, it can be applied to more fields and domains. Next to robotics and computer vision, NLP is a domain that can profit from RL approaches for summarization, machine translation, text generation and more. [Li, 2017]

2.2.1 Reinforcement Learning in NLP

In Section 2.1.5 we looked at some RL approaches for text simplification, where it was mainly used to sidestep issues with simplification data, such as availability or quality of data. Other work showed that RL can be used to solve different NLP problems.

Several approaches have tackled MT with RL, but these have been lacking behind supervised methods. MT problems have a very large action-space, being the size of the vocabulary $\mathcal{V}$ for producing a single word, or the product of $\mathcal{V}$ for each word in a sentence. With the reward being relatively sparse and the low number of possible correct sentences, it can be pretty inefficient to train an MT model that holds up with the performance of supervised approaches. [Choshen et al., 2019] There are sufficient large-scale datasets for many language pairs, which have been shown to improve the translation quality of models [Wu et al., 2018].

As already stated, summarization is a task closely related to text simplification. There have been various approaches using RL to learn extractive or abstractive summarization:

Narayan et al. [2018] presented an approach for extractive summarization. They extract individual sentences and learn to rank the sentences for summary generation according to the ROUGE metric, an evaluation metric for summarization [Lin and Hovy, 2003]. Their model learns this objective using the REINFORCE algorithm.

REINFORCE by Williams [1992] is a policy gradient algorithm in RL. The key idea is that an action from a probability distribution is selected based on a model's policy, e.g. a word is generated based on a conditional distribution over the vocabulary. Based on the agent's action, the objective is to increase the probability of actions that return a higher reward and decrease the probability of bad actions. With this, a gradient needs to be estimated to update the parameters of a model with gradient ascent.

This estimation can be done in different ways, like learning an estimated baseline reward that can be subtracted from the estimated future reward of the model's output [Nakamachi et al., 2020].

Laban et al. [2021a] tackle abstractive summarization with the use of self-critical sequence training, a form of the REINFORCE algorithm, which relies only on the model's outputs to learn. This method is further explained in the following section.

2.2.2 Self-critical Sequence Training

Self-**c**ritical **s**equence **t**raining (SCST) was introduced by Rennie et al. [2017], where they used RL to optimize a image captioning system and achieved state-of-the-art results with their training approach. SCST is a actor-critic method based on the REINFORCE algorithm, but rather than estimating the reward signal or how the signal should be normalized, it utilizes the current model's outputs to normalize the rewards. This is achieved by generating two candidates: First, a sequence of words $w^S = (w_1^S, ...w_T^S)$, where each word w_t^S is sampled from the model's probability distribution. Secondly, a sequence as baseline $\hat{w}$ from the current model is generated by taking the arg max of the model's word distribution at every time-step t, also referred to as *greedy decoding*.

The idea of SCST being self-critical is to use the reward of $\hat{w}$ as a baseline at test-time for the current model instead of training a second model to estimate the expected future rewards as a baseline. Actor-critic methods rely on this second critic-network, which estimates the reward function rather than the actual rewards.

The loss of the negative reward of a sample w^s is:

$$L = (\hat{R} - R^S) \sum_{i=0}^{N} \log p(w_i^S | w_{<i}^S, P) \tag{6}$$

where $\hat{R}$ represents the reward obtained by the current model and R^S the reward computed for the sampled sequence. $p(w_i^S|...)$ represents the probability of the i-th word conditioned on the sequence of length N generated so far and an input paragraph P. If the reward of the sampled sequence R^S is higher than $\hat{R}$, the model is optimized towards the sampled sequence and its probability is increased. Otherwise, if the reward of the sampled sequence is lower than the baseline, its probability is decreased.

This method was later successfully extended to other text generation tasks, such as summarization [Laban et al., 2021a; Celikyilmaz et al., 2018] and question generation [Zhang and Bansal, 2019].

In Laban et al. [2021b] an improvement of SCST was presented, which will be used in this work. One limitation with SCST is when the rewards of the sample and baseline are comparable, and the loss is nearly zero, which gives the model very little to learn. The extension proposed is called *k-SCST*, where $k > 2$ sampled candidates are generated, and each candidate's reward is computed. The mean reward $\bar{R}^S$ of the candidates is used as a baseline instead of $\hat{R}$, making the loss

$$L = \sum_{j=1}^{k} (\bar{R}^S - R^{Sj}) \sum_{i=0}^{N} \log p(w_i^{Sj} | w_{<i}^{Sj}, P). \tag{7}$$

The higher an individual candidate's reward R^{Sj}, the more it contributes to the loss and therefore influences the optimization of a given model. Increasing the number of candidates k leads to the rewards becoming more stable and the average reward higher. [Laban et al., 2021b]

Chapter 3

GUTS

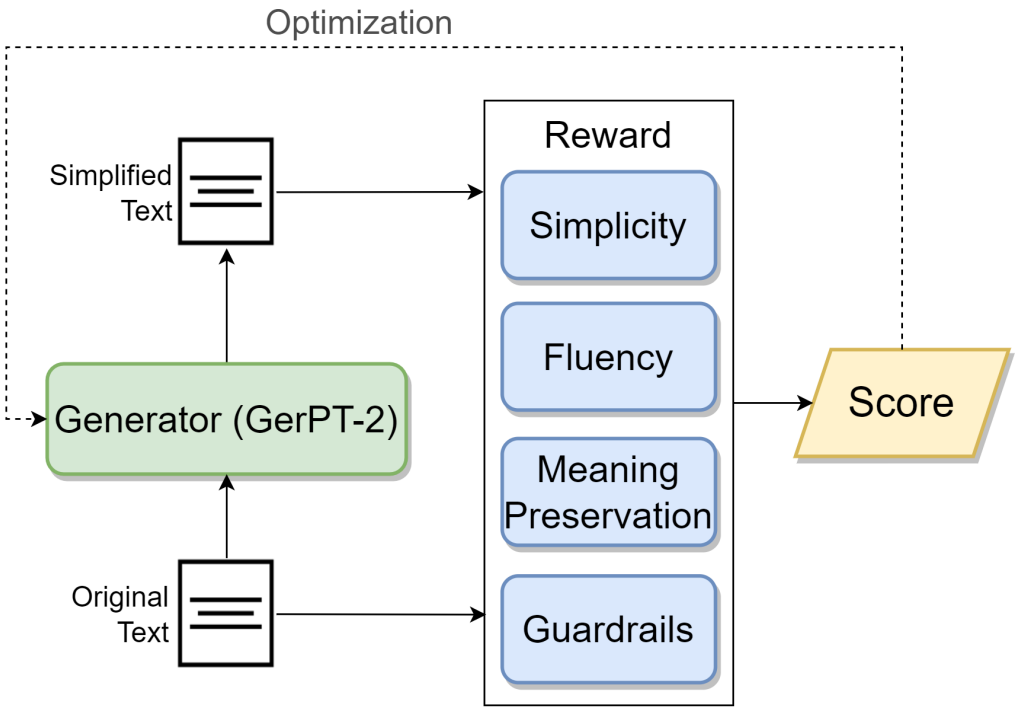


Figure 2: The GUTS Training Framework

In this chapter the ATS approach, named GUTS, is presented. Figure 2 visualizes the training framework.

First, the reward used as a training objective will be presented. The reward is split into individual scores for the components simplicity, meaning preservation, fluency and guardrails. Each score rates the produced simplifications, based on different criteria.

Second, the generator model that produces the simplifications is described. Here the training with the algorithm k-SCST and the sampling of different simplification candidates from the generator are explained.

3.1 Reward

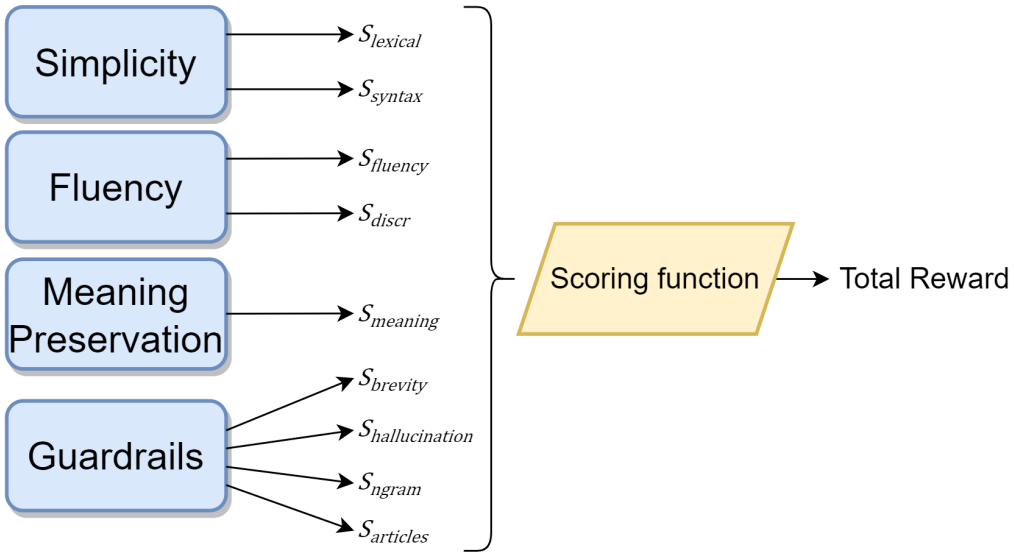


Figure 3: The GUTS reward composition

The approach used in this work, from Laban et al. [2021b], can be described as a non-differentiable reward maximization problem. For every original paragraph, P and a corresponding generated simplification S, different scores in the range of $[0, 1]$ for simplicity, meaning preservation, fluency and guardrails are received. These individual reward scores are then combined into a single reward using a scoring function. During the first experiments, a product scoring function that was described in the KiS paper [Laban et al., 2021b, §3], was used. With this, the product of all scores is used as the total reward. This has the advantage that the guardrail scores can be used to automatically set the total reward to zero. The guardrail scores serve as penalties and return a value of either 0 (fail) or 1 (pass). For example, if a simplification is too short to be considered, the brevity score $S_{brevity}$ is set to 0 and consequently, the whole reward too. The author of the KiS paper Phillipe Laban, reported better results by using the *logsum* of the scores for a scoring function.[1] This function offers the possibility to assign a weight to each score of the reward. The scoring function used in this was adapted according to these insights:

$$R = \sum_{i=0}^{N} W_i \log(s_i) \tag{8}$$

where R denotes the total reward for a simplification. N is the number of individual scores of the reward, and s_i describes an individual score with its assigned weight W_i. This way, not every score has the same impact on the overall reward. A drawback of this scoring function is that the guardrail scores cannot zero the score. To work around this, these scores are either set to 0.0001 or 0.9999.

To analyse the reward's scores, KiS used the 40 000 paired simple-complex paragraphs from the Newsela dataset. This dataset consists of news articles. Since there exist no

[1] This information was retrieved through E-Mail communication with Phillipe Laban.

Original	**Simplification**
Sie leisten sinnvolle Arbeit, entwickeln soziale Kontakte und Kompetenzen. Dabei achten wir immer darauf, Ihren Wünschen an einen Arbeitsplatz, Ihren Fähigkeiten und Neigungen zu entsprechen. Bei uns finden Sie angepasste Arbeitsplätze und Vorrichtungen und erhalten Ihre nötige persönliche Unterstützung und Hilfe. Die GWW hilft Ihnen gesellschaftliche Teilhabe und individuelle Selbstbestimmung zu realisieren. Wir bieten Ihnen umfassende Qualifizierungs- und Bildungsangebote. Unsere Jobcoaches unterstützen Sie auf Ihrem Weg auf den Arbeitsmarkt. Alle Menschen sollen gemeinsam leben und arbeiten können: Das ist unsere Vision.	Das Ziel der GWW ist: Alle Menschen sollen gemeinsam leben und arbeiten können. Bei der GWW gibt es angepasste Arbeitsplätze für Menschen mit Behinderung und Menschen mit einer psychischen Erkrankung. Die Job-Coaches unterstützen Sie auf Ihrem Weg auf den Arbeitsmarkt.

Table 2: Example article from the GWW dataset

German simplification datasets of this size, two datasets have been used as references: The *TextComplexityDE dataset* and a manually collected dataset of parallel articles from the website "Gemeinnützige Werkstätten und Wohnstätten" [Gemeinnützige Werkstätten und Wohnstätten - GWW, 2022]. The latter is referred to as the *GWW dataset* in this thesis.

The GWW dataset was manually created for this work by aligning original articles with their simplified versions from the website. The dataset consists of 52 parallel articles, mainly texts for disabled people containing information and help about topics like work or living. While this dataset contains simpler simplifications than TextComplexityDE, the information contained in the complex article and its simplification differs more.

TextComplexityDE dataset contains 23 articles from three different domains in Wikipedia. These articles are split into individual sentences with the corresponding simplification written by humans. It holds a total of 1019 sentences. Each simplification was rated for complexity, understandability and lexical difficulty. [Naderi et al., 2019]

Since this approach aims to simplify on a paragraph-level, the individual sentences from the same Wikipedia articles were combined to form paragraphs. The composed dataset resulted in 23 different Wikipedia articles. On some occasions, the sentences in the composed articles were not logically sequential. The composed TextComplexityDE dataset is used as a reference, because it showed good simplification phenomena and retained most of the content and information from the original text. With the GWW dataset more information from the original article got lost in the simplification. While the GWW dataset is also used loosely for designing the rewards, TextComplexityDE will serve as the primary reference. Since TextComplexityDE uses sentences from Wikipedia articles and GUTS is trained on extracted paragraphs from Wikipedia, the dataset is

overall better fit than the GWW dataset.

In the next section, the individual scores forming the overall reward are described. First the lexical and syntactic simplicity scores are presented.

3.1.1 Simplicity

The simplicity scores should tell whether the generator's output is linguistically simpler than the original paragraph. The methods from KiS for the lexical and syntactic simplicity were used and adapted for the German language.

Lexical Simplicity

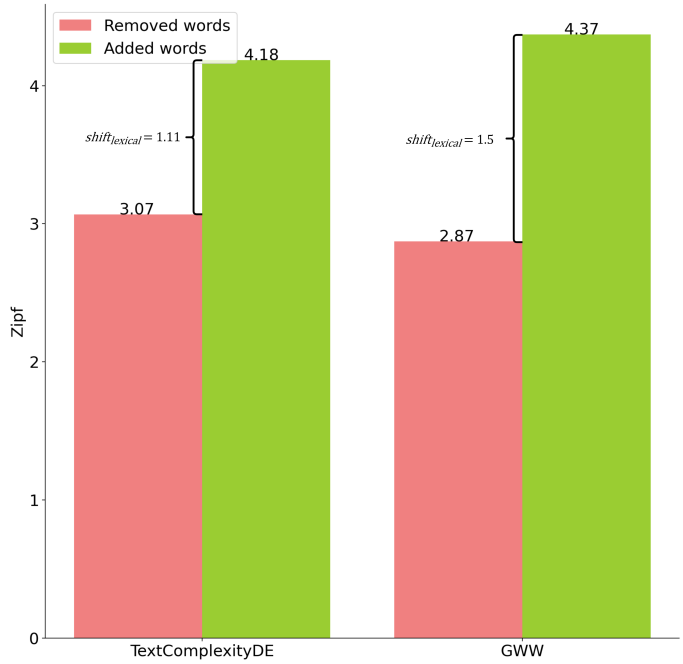


Figure 4: Plot of mean Zipf values for removed and added words

For determining lexical simplicity, the approach from Laban et al. [2021b] has been used. The objective is to replace complex words or phrases with simpler synonyms. KiS's score relies on the observation that word frequency and difficulty are correlated [Breland, 1996]: A word that occurs frequently in a language is assumed to be simpler to understand than a word that rarely appears. Since word frequencies follow the mathematical form known as Zipf's law [Li, 2002], the implementation by Speer et al. [2018] is used to determine the word frequency. With this, the Zipf frequency of a word is computed by taking the logarithm with base 10 of the number of times the word appears per billion words. The frequencies are adjusted on a $[0, 8]$ range with log normalization for all words. The frequency data is based on large text data from the websites such as Wikipedia, Twitter, Reddit and News data. E.g. the word "essen" (English: "eat") has a Zipf frequency of 5.33, while the related word "verzehren" (English "consume"/"eat") has a frequency of 3.2. In this case, "essen" is assumed to be simpler due to its higher frequency. The lexical

score $S_{lexical}$ builds upon these word frequencies to compute if the vocabulary difficulty changed between the original paragraph and the simplification.

Using this implementation, the lexical shift between the original paragraph and the simplification is computed. First, we strip all stop-words from the texts, as they should not be considered. Stop-words are very frequent words in a language but do not carry a significant semantic relevance for a text. They only have a grammatical or syntactic function. [Rajaraman and Ullman, 2011] Examples for stop-words in German are conjunctions (e.g. "und", "oder", "weil"), definite articles (e.g. "der", "die", "das") and indefinite articles (e.g. "einer", "eine") [Loper and Bird, 2002].

Next, all remaining words are lemmatized, to have a more accurate comparison between morphologically different words with the same base form. Two sets of words are created: One set contains all words that have been removed, and one set with words that have been added during simplification. The most complex or least frequent 15% of words from both of these sets are kept, all other words are filtered out. Then the average Zipf value for each set is computed: $\overline{Zipf}_{add}$ for the set of added words and $\overline{Zipf}_{rem}$ for the set of removed words. With these values, the lexical shift between S and P can be calculated:

$$shift_{lexical} = \overline{Zipf}_{add} - \overline{Zipf}_{rem} \tag{9}$$

The lexical shift or the difference between the $\overline{Zipf}_{add}$ and $\overline{Zipf}_{rem}$ should be positive for a successful lexical simplification. Figure 4 shows the average values for the most complex 15% added and removed words from TextComplexityDE and GWW. The lexical shift is positive for both reference dataset, showing lexical simplicity gains.

Algorithm 1 Calculation of the lexical score $S_{lexical}$ with a linear ramp function

$shift = shift_{lexical}(P, S)$
if $shift \leq target$ **then**
    $S_{lexical} \leftarrow shift/(target + 0.001)$
else
    $S_{lexical} \leftarrow 1 - 0.25 * (shift - target)/(target + 0.001)$
end if
$S_{lexical} \leftarrow max(0, min(x, 1))$ ▷ Clamp the score between 0 and 1

The final score $S_{lexical}$, which is integrated into the reward, is computed by dividing the lexical shift by a desired target shift. With a target value of 0.8 the TextComplexityDE corpus achieved a median score of 0.82. This value has been used for this work's experiments and training runs. KiS chose a target value of 0.4, according to their reference dataset [Laban et al., 2021b, §3.1.2].

The score is then clipped between 0 and 1 and has a ramp shape, where the score falls off when achieving a $shift_{lexical}$ above the target value. Algorithm 1 describes how the $S_{lexical}$ is computed from the shift.

Figure 5 presents a simplification example from the TextComplexityDE dataset, for which the lexical score is calculated. Complex words like "Streichriemen", "Privatbereich", or "abgeledert" are removed, while more frequent words "Lederriemen" or "abstreichen" are added during simplification. The average Zipf values are retrieved from the 15% of the most complex words from both texts. Then the lexical shift is computed from these values. According to the scoring function, the lexical score is selected. In this case, the

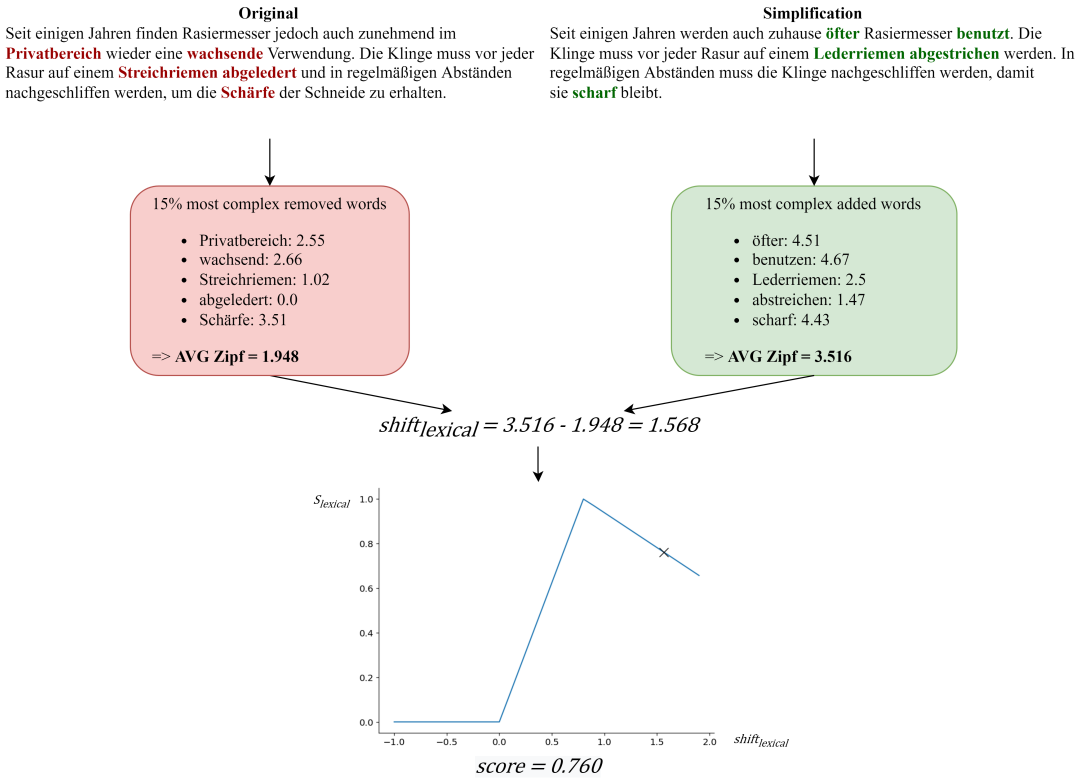


Figure 5: Example for the calculation of $S_{lexical}$

lexical shift was higher than the defined target of 0.8. A function that drops off after the target value is used to stop the generator from overshooting this score. Therefore a value below 1 is returned. Conducted experiments showed that always returning a score of 1 even when $shift_{lexical}$ is above the target led to undesirable behaviour of the generator.

Syntactic Simplicity

To measure the readability of their generator's output Laban et al. [2021b, §3.1.1] used the readability metric FKGL. Since there was no German adaption for this metric, the adaption FRE was chosen for this score instead. Short sentences with short words are scored well with these metrics. Further informations about FKGL and FRE are outlined in Section 2.1.3.

The objective is to reward the model for generating shorter sentences. For the syntactic score $S_{syntactic}$, the approach from KiS has been adapted: Laban et al. [2021b, §3.1.1] argue that an already syntactically simple paragraph should not require any further simplification and define the target FKGL conditioned on the original paragraph's FKGL score. They constructed their score according to the reference corpus Newsela. The lack of large-scale parallel datasets available for reference made it hard to closely adapt the score to the reference like they did in KiS, where they created a dynamic target based on the dataset Newsela.

Loosely based on the analysis of our reference datasets, a static target FRE of 60 was chosen for the score calculation. Figure 6 shows the average FRE values for the complex

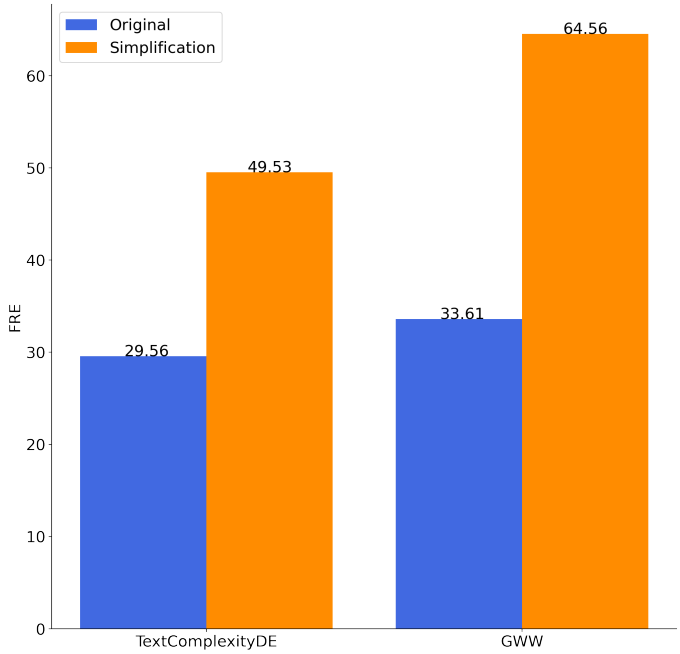


Figure 6: Flesch Reading Ease values of the reference datasets

and simple paragraphs from the references. This target corresponds to the 8th and 9th grade school level in the USA. The higher the FRE value of an original paragraph P, the less it needs to be simplified. If a paragraph has an FRE score of 60 and above, it does not need to be simplified syntactically, as it has already reached the target FRE and is assumed to be simple enough.

Algorithm 2 Calculation for the syntactic score $S_{syntactic}$ with a linear ramp function

$target \leftarrow 60 - FRE(P)$
$readability \leftarrow FRE(S) - FRE(P)$
if $readability \leq target$ **then**
    $S_{syntactic} \leftarrow readability/(target + 0.001)$
else
    $S_{syntactic} \leftarrow 1 - 0.25 * (readability - target)/(target + 0.001)$
end if
$S_{syntactic} \leftarrow max(0, min(x, 1))$ ▷ Clamp the score between 0 and 1

Algorithm 2 shows how $S_{syntactic}$ is computed. It also uses the same ramp-shaped linear function used with the lexical score to avoid the generator from overshooting the simplicity goal, which can cause problems with fluency and lead to worse readability.

Figure 7 shows a simplification example from TextComplexityDE. The original sentence is split up into two sentences in the simplification. The original text has an FRE value of 39.85, which corresponds to college-level reading difficulty. The simplification raises the FRE value to 66.55, which is slightly above the goal FRE value of 60. Since the computed readability value overshoots the target, it receives a score of 0.9. The syntactic score $S_{syntactic}$ then flows into the overall reward.

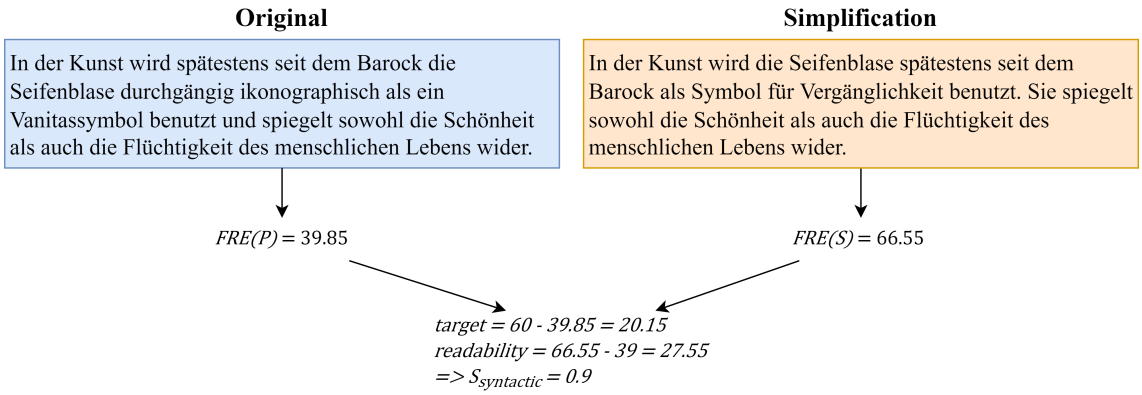


Figure 7: Example for the calculation of $S_{syntactic}$

3.1.2 Meaning Preservation

Besides the simplicity, it has to be ensured that the meaning of the original paragraph P is preserved as well as possible in the simplification S. In KiS, this is realized through an approach introduced by Laban et al. [2021a] for text summarization. Laban et al. [2021b, §3.3] adapt this so-called *coverage model* to the simplification domain. The coverage model is a transformer-based model, which is trained to look at the generated simplification and fills in masked words in the original text. The more masked words the model can correctly predict, the better the content is preserved in the simplification. In contrast to summarization, where only a limited number of the most important words are masked [Laban et al., 2021a], all non-stop words are masked for simplification. They fine-tuned a Roberta transformers model [Liu et al., 2019] on the coverage task and tested it on the Newsela corpus, where they achieved a coverage score of 0.76.

Unfortunately, these results could not be reproduced in this work. It was attempted to replicate their training approach by fine-tuning a German transformer model to predict masked words in German news articles from the MLSUM dataset [Scialom et al., 2020]. This dataset contains news articles and their corresponding summaries. These summaries and the articles first few sentences have been used as a substitution for simplifications. The rest of the article corresponded to P during training, and its keywords were masked accordingly. A German BERT model [Chan et al., 2021] was trained on this data to fill in the masked words correctly. Multiple runs with different masking rates and parameter settings have been tried, but the test results on the TextComplexityDE dataset were not sufficient. Therefore this scoring method for meaning preservation has not been used in this work. Moreover, this method does not consider the substitution of synonyms, because it uses exact matching of keywords to calculate the score. Since lexical simplification is often achieved by using simpler synonyms or paraphrases, other approaches might be more suitable to measure the meaning preservation in ATS. Future work can further experiment with this approach.

In this work, a different approach for the meaning preservation score $S_{meaning}$ is presented. First, the individual sentences from S are aligned to the most similar sentence from P. By aligning these sentences, operations like sentence splitting are considered. The aligned sentences are then scored with BERTScore [Zhang et al., 2019], to make use of contextual similarity between them and consider synonymity. Moreover, added infor-

mation in S, that was not in P is penalized, which the coverage model in KiS does not consider.

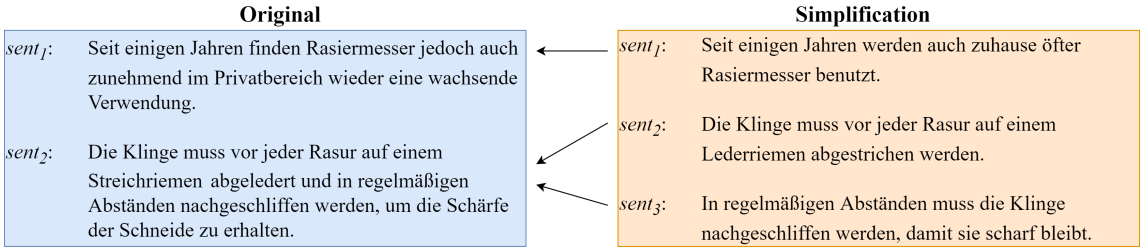


Figure 8: Sentence alignment example

Sentence alignment

First, the two paragraphs P and S are split into individual sentences. P is split into a set of sentences $sents^P = [sent_1^P, sent_2^P, ..., sent_n^P]$ and S respectively into $sents^S = [sent_1^S, sent_2^S, ..., sent_n^S]$. Next, a vector representation for every sentence in both sets is received. This is done by using a sentence transformer model based on [Reimers and Gurevych, 2019], which achieved state-of-the-art performance on sentence similarity tasks. For this, a model from the huggingface transformers library [Reimers and Gurevych, 2021] has been used, which showed a good trade-off between speed and performance.[2]

Each sentence vector from the simplified sentences $u^s = [u_1^s, u_2^s, ..., u_n^s]$ is aligned with the most similar sentence vector from the original paragraph $v^s = [v_1^s, v_2^s, ..., v_n^s]$. The similarity between the sentence embeddings is measured using the cosine similarity. Multiple sentences from S can be aligned to one sentence from P. An example is presented in Figure 8: $sent_2$ and $sent_3$ in the simplification are aligned to the same sentence $sent_2$ from the original paragraph.

When a sentence from S is not similar enough to any sentence from P, it does not get aligned. This threshold is set to a cosine similarity of 0.25. When the sentence is unaligned, it negatively impacts the score, as it is suspected to contain irrelevant information.

BERTScore

Next, BERTScore is used to compute the meaning preservation score $S_{meaning}$. BERTScore was initially tested on MT and image captioning to automatically measure the similarity score between two sentences. BERTScore fixes common problems of n-gram-based metrics like ROGUE or BLEU. These methods often fail to match paraphrases due to exact string matching or falsely punish reordering of sentences and paraphrases. Instead, BERTScore utilizes the contextual embeddings produced by a BERT model to compute the similarity between every token from the candidate and reference sentence. The contextual embeddings from BERT represent the words depending on the surrounding text and can capture phenomena like synonymity. [Zhang et al., 2019]

It has been previously described how the sentence alignments between P and S are retrieved. If multiple sentences were aligned to one sentence, these are then concatenated

[2]https://huggingface.co/sentence-transformers/msmarco-distilbert-multilingual-en-de-v2-tmp-lng-aligned

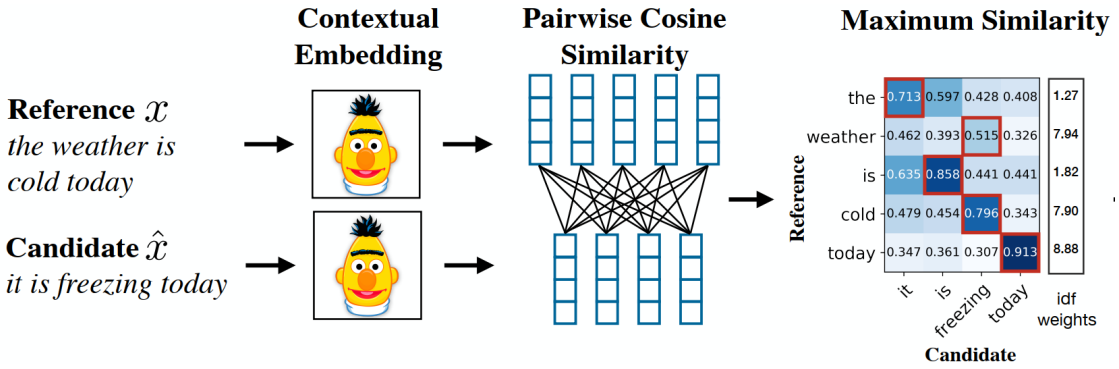


Figure 9: Illustration of the BERTScore computation with a candidate and reference sentence [Zhang et al., 2019]

and treated as a single sentence. Using the example from Figure 8: the sentences $sent_2$ and $sent_3$ from the simplification are combined to a single sentence, which is then used to calculate the BERTScore.

Let x be a reference sentence with n tokens $x = x_1, x_2, ..., x_n$ and a candidate sentence $\hat{x} = \hat{x}_1, \hat{x}_2, ..., \hat{x}_m$. A token is a meaningful sub-word. Tokenization is used to split the text into smaller parts for processing. Every token is represented as a contextual embedding by the BERT model. These embeddings are used to compute a similarity matrix between all tokens of x and $\hat{x}$ (see Figure 9). In the case of text simplification, the reference sentence x denotes the original paragraph and $\hat{x}$ consists of one or multiple combined sentences from the simplification. For the calculation of the final score, the F1 score is computed from the similarity matrix for each of the aligned sentence pairs.

$$R_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} x_i^\top \hat{x}_j \ , \ P_{BERT} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} x_i^\top \hat{x}_j \tag{10}$$

$$F_{BERT} = 2 \frac{P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}}. \tag{11}$$

In Equation 10 the part $x_i^\top \hat{x}_j$ is equivalent to the cosine similarity (see Equation 5): $\frac{x_i^\top y_j}{\|x_i\| \|y_j\|}$. Since BERTScore uses pre-normalized vectors, the calculation can be reduced to only the dot product between two vectors. Furthermore, greedy matching is used where each token is matched against the most similar token from the other sentence. The recall R_{BERT} captures how well information from the reference x is contained in the candidate $\hat{x}$, or in the case of simplification: How much of the contents from an original sentence are contained in the simplification. The higher this value, the less information is lost. The precision P_{BERT}, on the other hand, describes how much of the information of a simplification occurs in the corresponding original sentence. If this value is high, we can conclude that no additional or unnecessary information was added to the simplification by the system. [Zhang et al., 2019]

For the BERTScore calculation the German BERT base model from the huggingface library is used [Chan et al., 2021].

Firstly, it has been experimented with using BERTScore on the whole paragraphs of text without the sentence alignment step. This method gave the generator more room

to cheat the score. Generated simplifications that used similar words to the original paragraph achieved a high score, even when the words were arranged in a completely different order so that the meaning of the text changed. By measuring the semantic similarity with already related sentences, this problem was solved.

Score calculation

For every sentence of P, the F1 BERTScore is computed for the aligned sentences of S. The formula for the meaning preservation score is as follows:

$$S_{meaning} = \frac{\sum_{(sent^P, sent^S) \in aligned} F_{BERT}(sent^P, sent^S)}{|aligned| + |unaligned|} \tag{12}$$

where *aligned* denotes the set of aligned sentence pairs from the original paragraph and the system's simplification, with the original $sent^P$ and simplified sentence $sent^S$. The sum of the F1 BERTScores of each sentence-pair is divided by the count of aligned sentences $|aligned|$ and the number of unaligned sentences $|aligned|$. The set of unaligned sentences contains original and simplified sentences that were not semantically related to another sentence. Sentences from P that had no matching simplification sentence are penalized because it is assumed that information got lost during simplification. Unaligned simplified sentences that were not semantically related to any original sentence are also punished since they are assumed to contain unnecessary or hallucinated content.

In the simplification process, sentences are sometimes removed entirely or can be added to further explain a concept or word. This behavior was unwanted in this work because neural text generation models are prone to hallucinate content [Cao et al., 2018]. The generator should have as little space as possible to cheat, the score $S_{meaning}$ was designed to preserve as much meaning of P as possible.

After the calculation, the maximum score was set to 0.8. All scores above this value were shifted to a score of 1. The TextComplexityDE dataset achieved an average score of 0.9 on its simplifications, the GWW dataset only 0.4 since it often has some large chunks of information removed for their simplified articles.

3.1.3 Fluency

Fluency, also referred to as naturalness, grammaticality and readability, is a property of a text that would be perceived as natural by humans [Kann et al., 2018]. Because the outputs of the generator should be as human-like as possible, two components to measure the fluency of simplifications are used: a pre-trained language model to measure the fluency in a deterministic way and a dynamic discriminator, which is iteratively trained to distinguish between original texts and simplifications. Both of these scores were also used in KiS to measure the fluency [Laban et al., 2021b, §3.2] and are slightly changed in this work to rate the fluency for the German simplifications.

Language-Model Fluency

Understanding the humans linguistic abilities has been a subject of debate since the 1950s in cognitive science, linguistics and psychology. It has been questioned if well-formed or fluent sentences are a binary condition or of probabilistic nature. Lau et al. [2017] argues

that the grammaticality of a text can be measured by observing its probability. In their work, they show that grammaticality can be modeled with a language model. Further studies support this finding, showing that language models can be used to predict the fluency of texts generated by a system for tasks like MT and other text generation tasks [Kann et al., 2018; Salazar et al., 2019].

Masked language models, such as BERT [Devlin et al., 2018], train by predicting randomly masked words in a sentence. The prediction for a masked word is represented by a probability distribution over the vocabulary $\mathcal{V}$ of a model. The loss between an actual word and a model's probability distribution is used to tell how probable the word in the given context is.

This can be used to measure fluency. Here every word is masked, and a model predicts each word based on the past and future words from the given text. The sum of the resulting log probabilities of every word w_t are then divided by the number of total words, to normalize different text lengths:

$$LM(w) = \frac{\sum_{t=1}^{|w|} \log p(w_t|w_{\setminus t}; \Theta)}{|w|} \tag{13}$$

where $LM(w)$ describes the language model probability of a word sequence $w = (w_1, w_2, ..., w_{|w|})$. The log probability for every word $\log p(w_t|\cdot)$ is conditioned on all other words in the sequence $w_{\setminus t}$ and the model parameters Θ. [Salazar et al., 2019] This way, the overall likelihood of a original paragraph $LM(P)$ and the generated simplification $LM(S)$ can be obtained.

Laban et al. [2021b] uses this approach to measure the fluency in KiS, by taking the likelihood of the original and simplified paragraph to construct a score:

$$S_{fluency} = \left[\frac{\lambda + LM(P) - LM(S)}{\lambda}\right]^{+} \tag{14}$$

, where $LM(P)$ and $LM(S)$ stand for the likelihood of the original and simplified paragraph obtained by a masked language model. If the loss of a generated simplification $LM(S)$ is higher than $LM(P)$ by λ or more, $S_{fluency} = 0$. The score is clipped between 0 and 1; if $LM(S)$ is above or equal to $LM(P)$, the score is 1 otherwise the score is a linear interpolation between 0 and 1. [Laban et al., 2021b, §3.2.1]

For this work, a German BERT base model from huggingface [Chan et al., 2021] was used. Laban et al. [2021b, §3.2.1] utilized a GPT-2 model instead. Despite being trained on less data, Salazar et al. [2019] have shown that BERT can match or even outperform GPT-2 on fluency of English texts.

Previously, Laban et al. [2021a] highlighted the importance of fine-tuning the model on the target domain to capture the fluency. This way, the generator is rewarded for outputs that are similar to the target domain. There was no information about fine-tuning their GPT-2 model in KiS [Laban et al., 2021b]. In this work, better results and less mistakes with the fluency model have been observed after fine-tuning the BERT model on German Wikipedia articles [Foundation, 2022]. The model was trained for roughly two hours using AdamW [Loshchilov and Hutter, 2017] as an optimizer with a learning rate of 10^{-5} and a batch size of eight examples. AdamW is a stochastic optimization method that builds upon Adam [Kingma and Ba, 2014], by decoupling the weight decay from the gradient update. Additionally, Automatic Mixed Precision (AMP) was used for training

[Micikevicius et al., 2017]. Mixed precision methods allow to train neural networks using a combination of single-precision floating points (FP32) and half-precision floating points (FP16), thereby reducing the memory requirements and speeding up the training process while also maintaining model accuracy. Both AdamW and AMP will be used for all other training tasks performed in this work.

A value of $\lambda = 0.12$ has been used since the TextComplexityDE simplifications reached an average score of 0.96 with this value. The GWW dataset has only gotten an average score of 0.49. This might be because the sentences in TextComplexityDE are from Wikipedia articles, which is the target domain the language model was trained on. The articles from GWW, on the other hand, are written in a different linguistic style. The λ value can be further fine-tuned in the future.

Original	**Simplification**
Die Gebiete des heutigen Belgiens wurden von 57 bis 51 v.Chr. durch Julius Caesar erobert. Der Name Belgien geht auch auf ihn zurück, der allen keltischen Stämmen nördlich der Flüsse Sequana (Seine) und Matrona (Marne) die Bezeichnung Belgae gab ("Galliorum omnium fortissimi sunt Belgae", De Bello Gallico, liber primus).	das den heutigen Belgiens wurden durch 57 bis 51 v von die die den den erober. Der Namen ist die die haben auch auf sie zurück, der alle keltischen Stämme nördlich der Flüsse Seqana (Ihre) und die Matrona (Die) die die die die die das Wort Belgae geben.

Figure 10: Incoherent simplification reaching a fluency score of **0.96**

Unfortunately, adapting this approach to the German language came with novel problems: It seemed to encourage shorter and more probable words, especially articles like "der", "die", "das" (English: "The"). Figure 10 shows an example from a simplification sample during one of the experiments. This instance reached a fluency score of 0.96. This might be because articles are relatively frequent words and therefore overall very probable in German, which results in a smaller loss. This issue was further explored and it was found that just adding repeating articles to a text, often decreases the overall loss of a text, therefore scoring it as more fluent.

This particular issue has been fixed by introducing a article repetition penalty (Section 3.1.4), that detects and penalizes repetitions of articles in a generated text.

While this fluency score sometimes had trouble penalizing certain ungrammatical behavior, it did recognize various grammatical error's and faulty punctuations. In the scope of this work, grammatical issues with this score have not been further explored. There may be various reasons for such errors, like the sub-optimal choice of a language model or an unsuitable λ value, due to the small reference corpus. Moreover, since German grammar and morphology is arguably more complex than English, this method might not be as feasible for rating the fluency of German texts as for English. This task needs more research regarding the German language.

Discriminator Fluency

Following the work of Laban et al. [2021b, §3.2.2] a dynamic discriminator was incorporated. This score is inspired by the Generative Adversarial Network (GAN) framework. A generative model G produces data to capture a desired data distribution, and a discriminative model D estimates the probability if a sample has been generated by G or came from training data [Goodfellow et al., 2014]. In this case, G is the generator that simplifies the examples and the discriminator D tries to predict if a given paragraph is a

generated simplification or an original paragraph written by a human.

The LM-Fluency score described in the previous section can be limiting as it is static and deterministic. Therefore it can be exploited by the generator (Figure 10). To counter this, the discriminator is dynamically trained with the generator.

During the generator's training process, both the simplification outputs and the original paragraphs are added to the discriminators training buffer. The original paragraphs are assigned a label of 1, and the generator outputs a label of 0. When the buffer reaches 4000 samples, the discriminator is trained with the data, and the buffer is emptied again. A German BERT model is trained using 90% of the training buffer for the discriminator. The discriminator is trained for five epochs. The end of each epoch is used as a checkpoint, where the discriminators model is saved along with the F1 performance tested on the last 10% of the training buffer. The best model of the five checkpoints is kept as the new discriminator until the training buffer reaches 4000 samples again. The model is trained using AdamW [Loshchilov and Hutter, 2017] as an optimizer with a learning rate of $10^{-}5$, a batch size of 6 and AMP.

KiS used a RoBERTa model [Liu et al., 2019] for this and retrained their model already after 2000 samples.

The discriminator's output probability for a simplified paragraph S is used to construct this score:

$$S_{discr} = p_{discr}(Y = 1|X = S) \tag{15}$$

With this dynamic approach, the generator attempts to maximize the likelihood of its outputs to appear human-like, while the discriminator learns to distinguish between generated and authentic paragraphs. [Laban et al., 2021b, §3.2.2]

3.1.4 Guardrails

In this section, the guardrail scores are described. These penalize common generator problems. Unlike the reward scores presented so far, which are continuous, the following guardrail scores are designed to be discrete and either succeed (1) or fail (0).

Brevity

The brevity guardrail is a score that ensures that the length of a generated simplification falls into the range of the original paragraph. The implementation from KiS [Laban et al., 2021b, §3.4.1] was used. The length of a text $L(\cdot)$ is calculated by counting the number of characters and is used to calculate the compression rate:

$$C = \frac{L(S)}{L(P)} \tag{16}$$

The brevity score returns 1 if $C_{min} \leq C \leq C_{max}$, otherwise it returns 0. In KiS values of $C_{min} = 0.6$ and $C_{max} = 1.5$ have been used. For this approach $C_{min} = 0.6$ and $C_{max} = 1.3$ have been used. With these settings all parallel articles from TextComplexityDE reached a score of 1, which we used as reference. In the GWW dataset only 21 of 52 articles fulfilled this criterion. As already stated, texts from this dataset often excluded parts of the complex articles in their simplification. For future work these values can be adapted.

Hallucination Detection

A common problem for text generation tasks like ATS or summarization are factual inconsistencies. An important requirement for these tasks is that the facts from the generated text matches the source text. [Fischer, 2021] Recent studies have shown that around 30% of summaries generated from neural models contain facts that do not occur in the input text [Cao et al., 2018; Falke, 2019].

The overall goal is that facts expressed in the simplification, were also present in the original paragraph. This aspect is also referred to as faithfulness [Cao et al., 2018]. While hallucinated facts can be factual [Maynez et al., 2020] it will not be considered in this work.

For hallucinations generated by the generator model, the score S_{hallu} is constructed for the reward. This score detects if new entities have been introduced to the simplification that did not appear in the original paragraph. The *inaccuracy guardrail* from KiS is adapted, which uses a Named Entity Recognition (NER) model on both paragraphs to check if an entity occurs only in the generated simplification. Even though new entities are sometimes introduced in simplifications written by a human, Laban et al. [2021b, §3.4.2] argue that they primarily observed the introduction of inaccurate novel entities.

Instead of directly matching named entities, a different approach is presented: First, the named entities from the generated simplification S are extracted. Next, the BERTScore library [Zhang et al., 2019] is used to obtain the words from P with the highest similarity to each extracted entity. The approach from KiS needs to consider different morphological forms and spellings of the extracted entities. While their implementation offers a variety of rules to detect such changes, certain forms might be undetected and can be falsely flagged. The method proposed in this work tries to avoid this problem. Furthermore, BERTScore has shown to correlate with human faithfulness judgments in abstractive summaries [Koto et al., 2020]. I argue that some forms of entities that would have been penalized with the KiS approach are taken into account with this method.

The NER model used is a multilingual RoBERTa model from the huggingface transformers library [Adelani, 2021]. RoBERTa is an updated version from BERT, with the masking method performed during pre-training instead of masking before training [Liu et al., 2019]. The model is trained to recognize locations, persons and organizations. First, all named entities from the generated simplification are extracted. In Figure 11, "Martha-Maria" and "Frankfurt" are recognized as named entities.

Secondly, the contextualized vectors are received from the BERT model, and a similarity matrix is created between the original text and the simplification. For this, the same BERT model used for calculating the BERTScore in Section 3.1.2 for $S_{meaning}$ is employed.

Next, the similarity value from the most related word in the original paragraph is selected for each detected entity. This value is then compared against a threshold, which was set to 0.74 in this work. This value can be picked from the similarity matrix, as shown in Figure 11. The entity "Martha-Maria" consists of three tokens: "Marta", "-" and "Maria". The average of the highest similarity values for each entity token is compared against the threshold. In this case the entity "Martha-Maria" has an average BERTScore similarity of 0.979 since the same entity exists in the original paragraph. The word "Frankfurt" is flagged as hallucinated, since its highest similarity ("Münchner": 0.508) is below the threshold. In this example, a hallucination was detected and the score

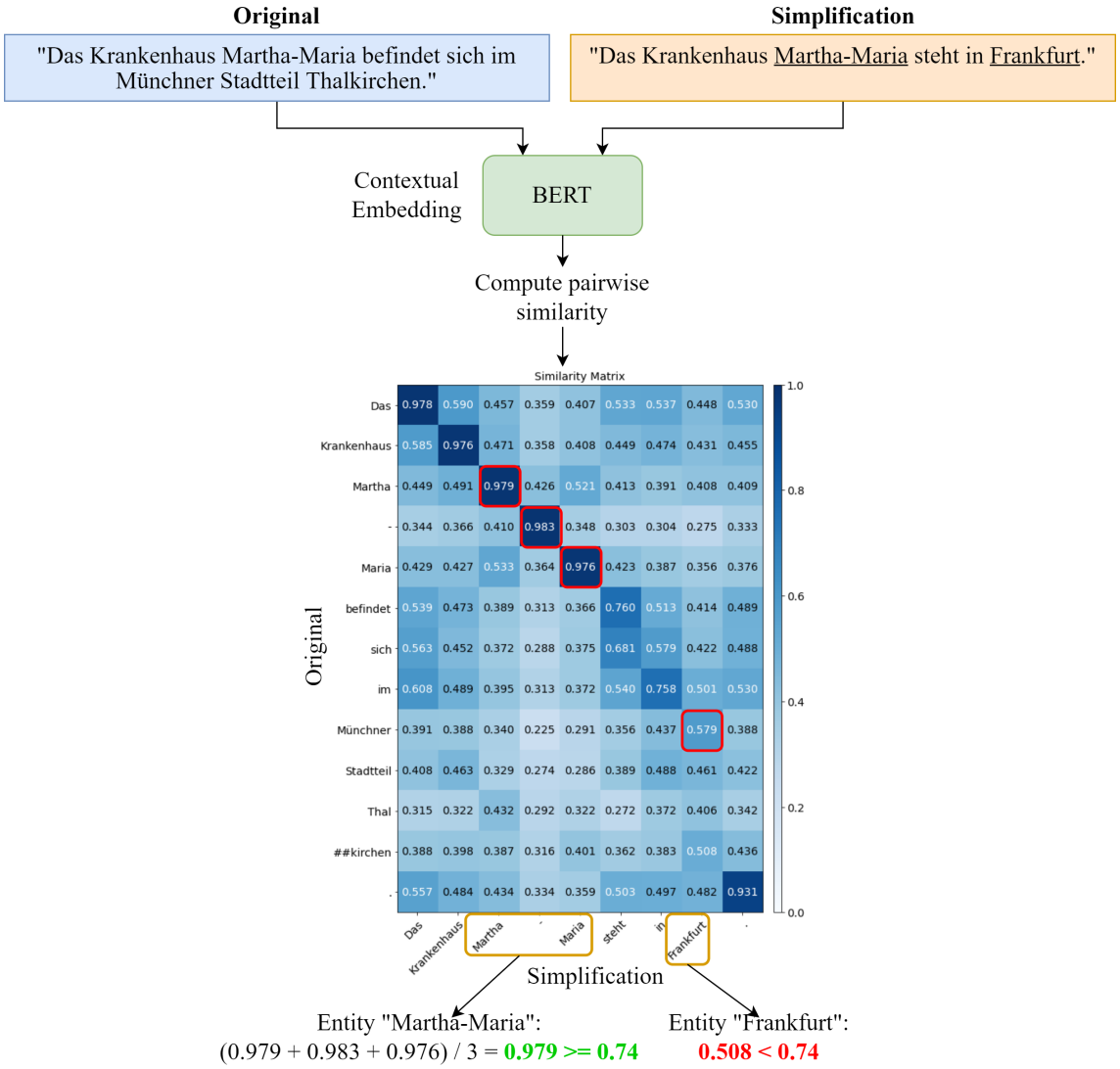


Figure 11: Hallucination detection algorithm

S_{hallu} is set to 0, if otherwise no hallucination is found a score of 1 is returned.

During the experiments, it was found that this method could identify entity hallucinations by the generator consistently. Still this method has a lot of room to improve, because factual inconsistencies were still one of the main problems with the generated simplifications in this work. For example wrong dates and numbers could be found regularly in the system's output. With the KiS model, this problem is experienced less, because their NER model detected numeric values such as dates, money, or percentages.

This approach and KiS can not detect other types of hallucinations, such as intrinsic hallucinations. An intrinsic hallucination describes the case when the generated text contains terms and concepts from the source but misrepresents the information and makes it unfaithful [Maynez et al., 2020]. The first entry in Table 3 shows an example of an intrinsic hallucination marked in red. The remaining table entries show problems with numeric values.

Original	Simplification
Werner Hirsch entstammte einer jüdischen Familie des gehobenen deutschen Bürgertums. Seine Mutter Helene Kallmorgen kam aus der Familie "von Bismarck". Sein Vater Walter Hirsch war Richter am Landgericht.	Werner Hirsch entstammte eine jüdische Familie des gehobenen deutschen Staatsten. Seine Mutter Helene war Richter am Gericht.
Die Wanzen werden 3,5 bis 4,5 Millimeter lang.	Die Wanzen werden drei bis 4,5 bis 5,5 Millimeter lang.
Im April 2014 gründete beide ein Textbüro.	2012 gründeten beide Textbüros.

Table 3: Examples of factual inconsistency from generated by GUTS

For future work, approaches like question answering to evaluate factual consistency [Wang et al., 2020] or using a post-editing corrector to identify and correct factual errors in generated text [Cao et al., 2020] would be interesting to explore for ATS.

Repeat N-Gram Penalty

Another guardrail score introduced in KiS is a n-gram penalty score, that checks if a 3-gram of words occurs more than once in the generated text. If the repetition of the same three sequential words is observed more than once in a generated text the score is set to 0, otherwise it remains 1. [Laban et al., 2021b]

A common problem with text generation models like GPT-2 is that they are prone to get stuck in repeating the same words [Holtzman et al., 2019]. With the score S_{ngram}, simplifications that contain repetitions are scored lower to minimize the probability of the generator of producing such repetitions.

Article Repetition Penalty

The last guardrail score introduced in this work detects and penalizes the repetition of German articles like "der", "die", "das". This score was not used in KiS and was only introduced for this approach. An example for this problem is found in Figure 10. As already stated in Section 3.1.3, this behavior was observed during some experiments, where the generator learned to cheat the fluency by repeating such high probable articles.

The score is set to 0 if three or more articles appear in a sequence, else it returns 1. With the introduction of this score, this behavior was unlearned from the generator. This will be further outlined in Section 4.3.

3.2 Generator

The following section presents how the generator is trained to simplify paragraphs. First, the properties of the generator model are described. Secondly, the sampling of the simplification candidates is explained. Here different decoding methods for sampling are outlined.

3.2.1 Learning

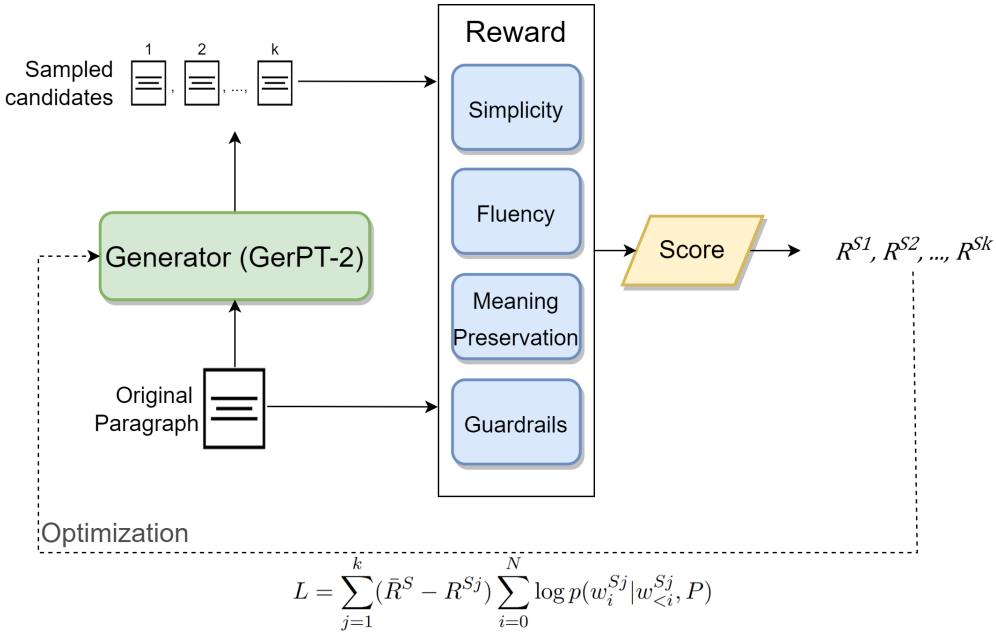


Figure 12: The GUTS Learning with k-SCST

Figure 12 displays how the generator learns: First k simplification candidates are sampled from the generator model, conditioned on an original paragraph. These candidates are then scored according to the reward, described in Section 3.1. From the resulting rewards $R^{S1}, R^{S2}, ..., R^{Sk}$ the mean reward $\bar{R}^S$ is calculated as a baseline for the loss. The loss L is computed with the difference of $\bar{R}^S$ and the reward of each candidate. The higher an individual reward of a candidate R^{Sj} is above the baseline, the more it influences the loss and contributes to the optimization of the generator. $p(w_i^{Sj}|...)$ describes the probability of the generator producing the i-th word w_i^{Sj} conditioned on the input paragraph P and the previously generated words $w_{<i}^{Sj} = w_1^{Sj}, ...w_{i-1}^{Sj}$.

For the generator model, Laban et al. [2021b, §4] use a pre-trained medium GPT-2 model. Since this work tackles the simplification of German paragraphs, a German GPT-2 model was used. This version, GerPT-2 [Minixhofer, 2020] was retrieved from the huggingface transformers library [Wolf et al., 2020]. The model was initialized using the weights of the English GPT-2, then pre-trained on 67GB of German texts from the CC-100 Corpus [Wenzek et al., 2020].

As an input the generator GerPT-2 gets a original paragraph P. First this text needs to be tokenized by GerPT-2's tokenizer. Generally speaking tokenization is the process of splitting a text into smaller parts. With GPT-2 and most other language transformers, the tokenizer splits the input text not only into words, but meaningful sub-words. E.g. the word "glücklich" is tokenized as a single word, "Textvereinfachung" on the other hand results in three tokens: "Text", "##verein" and "##fachung". The symbols "##" signal, that this token belongs to the preceding one.

Given an input, the generator produces one token at a time and adding it to the input sequence until the **E**nd-**o**f-**S**entence (EOS) token is generated. Since GPT-2 consists only

of transformer decoder blocks, the model is not aware what the initial input was and which words the model itself generated, or in the case of text simplification: where the original paragraph P ends and where the produced simplification S begins. A separator-token ($< |sep| >$) is introduced to the generator and its tokenizer to mark this position in the sequence. This token is added to the end of P to signal where S begins.

Before the training process, the generator first gets fine-tuned on a *copy task*. Using this task, the generator learns to output an exact or close copy of the input. This is a good baseline to start the simplification process. Also, the generator learns the purpose of the introduced separator-token. The training script from the Summary Loop Github repository has been used for this training task [Laban et al., 2021a]. The generator was fine-tuned with AdamW [Loshchilov and Hutter, 2017], a learning rate of $2 \cdot 10^{-5}$, a batch size of eight examples and AMP. The model was trained on this task for about 25 minutes.

When the generator was trained for too long on the copy task, the sampled simplification candidates during simplification training were often too similar or even an exact copy of the original text. This low diversity resulted in very similar rewards, which limited the training signal for the generator.

3.2.2 Candidate Sampling

For training with k-SCST, k simplification candidates for one original paragraph are sampled from the generator every training step. Since the sampled candidates define the quality of the training data, choosing a decoding strategy is an important decision.

A probability of a language model for a sequence of words is described as the product of its conditional probability distributions over all words in a vocabulary:

$$p(w) = \prod_{T}^{t=1} p(w_t|w_{<t}) \tag{17}$$

where T denotes to the length of the word sequence w and $p(w_t|w_{<t})$ describes the probability distribution for the next word w_t, conditioned by the previous sequence of words $w_{<t}$. A decoding strategy sets the rules for selecting a word w_t from a language models probability distribution $p(w_t|w_{<t})$. [Meister et al., 2022]

In the work of Laban et al. [2021b], there was no information about the decoding strategy used for sampling the simplifications during their training run. For this work, different methods have been tested. An overview of decoding strategies for natural language generation is presented in the following. Some have been tested for this work.

Greedy decoding is the standard way to generate text using generative language models, like GPT-2. At each step t the most probable word is selected.

$$w_t = \arg\max p(w_t|w_{<t}) \tag{18}$$

The next word w_t is chosen by selecting the word with the highest probability from the distribution $p(w_t|w_{<t})$.

Greedy decoding has issues with producing word or n-gram repetitions, generic or degenerate texts that seem not human-like. [Vijayakumar et al., 2016; Holtzman et al., 2019]

Beam Search tries to find the word sequence with the highest probability by tracking the most likely n beams at every step. Mathematically, the objective is to find the beam that maximizes Equation 17. At every time-step, the most probable sequences are chosen. One can find overall more likely sequences than greedy decoding with this method. [von Platen, 2020]

While Beam Search performs well for machine translation, it suffers from word repetitions and leads to degenerate texts in other language generation tasks. Holtzman et al. [2019] showed that high probability texts in language models are often generic, repetitive and awkward.

Random Sampling is a method where the next word is randomly picked on its conditional probability distribution: The higher the probability of a word, the more likely it will be chosen. This non-deterministic method can often result in incoherent text unrelated to the model's input. [Holtzman et al., 2019]

Random sampling can be improved by making the distribution $p(w_t|w_{<t})$ sharper. This is achieved by lowering the *temperature* in the softmax function:

$$q_i = \frac{exp(z_i/temp)}{\sum_j exp(z_j/temp)} \tag{19}$$

where q_i denotes the probability of a word from the vocabulary and z_i is the words logit vector produced by a neural network. Before calculating the softmax, the logit vector z_i is divided by the temperature *temp*. The softmax function normalizes all logit vectors between 0 and 1 at every time-step. [Hinton et al., 2015]
The probabilities of every word in the vocabulary $q = (q_1, ...q_n)$ make up the probability distribution $p(w_t|w_{<t})$. By lowering the temperature below a value of 1, the distribution becomes sharper, meaning that the likelihood for high probability words is increased and the likelihood for low probability words is decreased. This can help to make the text less random and incoherent. Increasing the temperature above 1 makes the distribution broader and therefore generates more random sequences. von Platen [2020]

Some experimenting has been performed with the temperature value. For training, an increase of the value to make the texts more diverse has been tried, but no improvements were noticed. For the evaluation, a temperature of < 1 was used to make the simplifications less random, which helps to generate more reliable simplifications.

Top-K Sampling builds upon pure sampling by filtering the K most likely words and redistributing the probability mass among the K remaining words. This decoding method was introduced by Fan et al. [2018] to generate more natural texts and to avoid the usage of improbable words, like with random sampling.

With $\mathcal{V}^{(K)} \subseteq \mathcal{V}$ being a set of the vocabulary $\mathcal{V}$ with the K most likely words the distribution is truncated and redistributed:

$$p^*(w_t|w_{<t}) = \begin{cases} p(w_t|p_{<t})/Z_t, & if\ w_t \in \mathcal{V}^{(K)} \\ 0, & else \end{cases} \tag{20}$$

with $Z_t = \sum_{w_t \in \mathcal{Z}^{(K)}} p(w_t|w< t)$.

While this method generates diverse and comprehensible results, choosing a suitable value for K can be difficult. With a peaked distribution, where most of the probability mass is concentrated into a few words, a suitable K value would be small. Using the same K value with a very flat distribution containing many moderately probable words would exclude some reasonable candidates. With a high K, a peaked distribution might include unreasonable words that result in gibberish texts. [Holtzman et al., 2019; Meister et al., 2022] This problem is visualized in Figure 13. Here the left plot shows a broad distribution, where reasonable words are excluded due to the filtering only the most likely five words. The right plot shows a peaked distribution, where the chosen K value is too large, and unsuitable words are included.

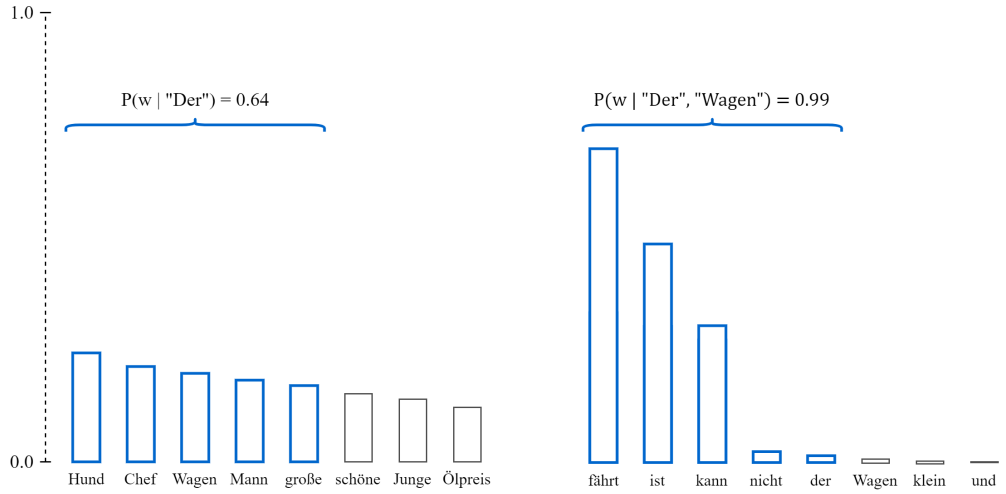


Figure 13: Peaked and flat distributions with Top-K sampling ($K = 5$)

Nucleus Sampling or top-p sampling was introduced by Holtzman et al. [2019] to fix the before mentioned issue with top-K sampling. Instead of sampling from the most likely K words, the smallest possible set of words is chosen, whose cumulative probability exceeds the value n. The probability distribution is filtered and redistributed in the same fashion as Equation 20, but with a different Z_t:

$$Z_t = \sum_{w_t \in \mathcal{Z}^{(n)}} p(w_t | w < t) \geq n \tag{21}$$

In the remainder of this thesis, the variable n will be referred to as p.

Figure 14 shows the same probability distributions as before with top-K sampling. This time, more reasonable words are selected in both the peaked and the flat distribution. In the flat distribution, more valid candidates are considered for sampling, and in the peaked distribution, unsuitable words are excluded.

Both nucleus sampling and top-K sampling work well on their own but can also be combined together. This combination allows to avoid low probability words, while also having a dynamic selection. [Holtzman et al., 2019]

Typical Sampling is a decoding strategy that is motivated by an informationtheoretic view of natural language. It is based on the assumption that humans use language

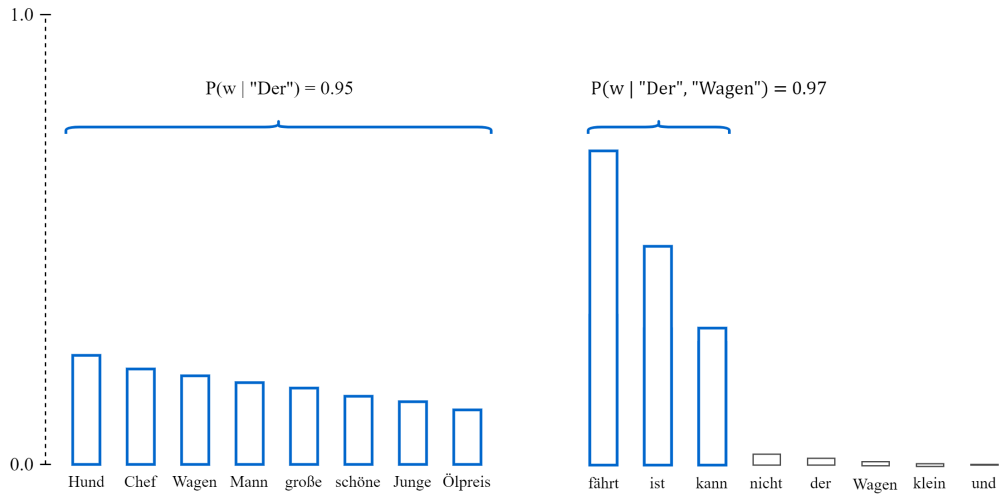


Figure 14: Peaked and flat distributions with nucleus sampling ($n = 0.95$)

to transmit information in an efficient manner. Because high probability words have a below average information content, for a text to be human-like the focus should be on transmitting information instead of choosing the highest probability word. [Meister et al., 2022]

This method computes the conditional entropy of the model at every time-step, based on the model's probability distribution over $\mathcal{V}$:

$$H(p(\cdot|w_{<t})) = -\sum_{w \in \mathcal{V}} p(w|w_{<t}) \log p(w|w_{<t}) \tag{22}$$

The conditional entropy represents the expected information content, given the prior sequence. The sampling distribution is then limited to words with a negative log-probability within an absolute range of τ, similar to n in nucleus sampling. First, $\mathcal{V}^{(\tau)} \subseteq \mathcal{V}$ is defined as the subset of words that minimize

$$\sum_{w \in \mathcal{V}^{(\tau)}} |H(p(\cdot|w_{<t})) + \log p(w|w_{<t})| \tag{23}$$

so that $\sum_{w \in \mathcal{V}^{(\tau)}} p(w|w_{<t}) \geq \tau$. Next the truncated distribution is renormalized equally to equation 20, and the next word in the sequence is sampled.

Typical sampling has been found to generate comparable or better quality texts on story generation and summarization tasks than nucleus sampling. [Meister et al., 2022]

The decision for a suitable decoding method depends on the text generation task. This can be roughly divided into open-ended, and directed text generation.

Directed text generation defines tasks that get an input text and output a transformation of this input. For these tasks decoding strategies such as greedy decoding or beam search generally work well. Examples are machine translation or text summarization.

Open-ended generation includes tasks like conditional story generation or text continuation. While the input constrains the space of acceptable outputs, it has a higher degree

of freedom of what word can come next. Holtzman et al. [2019] argues that high quality human language does not follow a language model's distribution of high probable words. Especially with longer texts, humans tend to prefer the use of more surprising words, while high probable generated texts are perceived as boring and predictable. [Holtzman et al., 2019]

Since ATS is similar to summarization, it falls under the directed text generation tasks. Nevertheless, for the training with k-SCST, multiple candidates with a certain degree of diversity are needed. I argue that the sampling for this case falls somewhere in between open-ended and directed text generation.

To sample the simplification candidates for training, a combination of nucleus and top-K sampling has been used. Random sampling and beam search were further experimented with, but both methods performed worse. Random sampling digressed from the topic of the original paragraph and beam search seemed to produce an increased amount of non-fluent and repeating texts. Moreover, typical sampling has been briefly tried for training, but it could not generate as diverse samples. Still, this has to be tested more extensively to be confirmed.

Evaluating decoding methods have been primarily researched for the English language, with very little attention on German. Further exploration and experiments with these methods for the German language and k-SCST are left for future work.

Chapter 4

Experiments

Multiple experiments and training runs of GUTS have been conducted for this thesis. Most of these experiments were used for testing functionality, settings and hyperparameters. These only ran for a few hours to one day. For comparison: Laban et al. [2021b] ran around 200 experiments with an average run-time of one week. Tests to that extent were not possible in this work due to time and resource limitations. It shows that much more experimenting with GUTS can be done to collect more insights and improve different components and settings. Furthermore, increasing the training time will improve the results.

In the following section, the data used for the experiments is described. Further on, the technical details and experimental setups for the two main training runs are explained. The resulting models of these runs are referred to as *GUTS-1* and *GUTS-2* hereafter. Lastly, some observations collected during experiments and training are described in this chapter.

4.1 Data

For training, a dataset of short paragraphs extracted from Wikipedia articles has been generated. The raw Wikipedia articles have been received from German Wikipedia dumps[1]. The dump from the 21st of January 2022 was downloaded and processed as follows:

1. The dump is preprocessed into articles using WikiExtractor [Attardi, 2015], a python script that extracts and cleans the texts from the dumps.

[1] https://dumps.wikimedia.org/dewiki/

Length	Count	AVG sentences	AVG words
Short (80-110 tokens)	539,921	3.6	70.5
Medium (110-140 tokens)	314,578	4.6	93.4
Long (140-175 tokens)	225,501	5.7	117.3
	1,080,000	4.6	93.8

Table 4: Training data used for the experiments

2. Empty and very short articles are removed.

3. Articles are split into individual paragraphs at each new line ("\n"), resulting in 10.8 million paragraphs. This way, it is ensured that the individual sentences of a paragraph are sequentially contiguous.

4. The paragraphs are further cut down into a length between 80 and 175 tokens. A dataset containing over one million short paragraphs has been used for training.

In step 4, the generator's tokenizer (see Section 3.2) is used to check the length of each paragraph. Even though the transformer models used for this approach can handle a sequence length of at least 512 tokens, the paragraphs are cut down to a maximum of 175 tokens. This has been done to speed up each training step and limit the GPU memory consumption by the models. A minimum of 80 tokens has been selected to see how well the generator learns to simplify texts with different lengths, while most paragraphs of 80 tokens still consist of multiple sentences. The data has been divided into three length categories, which are shown in Table 4. For the manual evaluation, 100 articles from each of the three length categories have been randomly selected.

Paragraph from 'Actinium': "Gibt man Natriumdihydrogenphosphat (NaH2PO4) zu einer Lösung von Actinium in Salzsäure, erhält man weiß gefärbtes Actiniumphosphat (AcPO4 · 0,5 H2O); ein Erhitzen von Actinium(III)-oxalat mit Schwefelwasserstoff bei 1400 °C für ein paar Minuten führt zu schwarzem Actinium(III)-sulfid (Ac2S3)."

Paragraph from 'Metaphysik': In einem "analogen Seinsverständnis" wird „Sein“ als das verstanden, was "allem" zukommt, wenn auch auf je verschiedene Weise ("Analogia entis"). Das Sein ist das, worin einerseits alle Gegenstände übereinkommen und worin sie sich zugleich unterscheiden. Dieses Seinsverständnis führt zu einer (dialektischen) „Seinsmetaphysik“. Der Gegenbegriff zum Sein ist hier das Nichts, da nichts außerhalb des Seins stehen kann. Sein wird hier als Fülle verstanden. Ein Beispiel für diesen Ansatz liefert die Spätphilosophie des Thomas von Aquin ("Summa theologica").

Figure 15: Difficult example paragraphs from Wikipedia

A challenge that arises with these paragraphs from Wikipedia is the linguistic diversity. The dataset presents a diverse range of topics. Furthermore, the articles are written in differing writing styles since there are many different authors who write and edit articles on Wikipedia. Having a diverse set of texts with niche vocabulary from a specific domain or scientific notations like in chemistry, biology, or mathematical articles is already difficult to simplify by a human. The first text in Figure 15, shows a paragraph from the Wikipedia article about "Actinium", where contextual and domain knowledge would be required to explain the abbreviations and concepts in the paragraph. The second example is from the "Metaphysik" article and shows logically complex formulations and interrelationships between terms.

Laban et al. [2021b, §A.1] used an unreleased corpus of 7 million English news articles as training data. Their training data contains arguably less linguistic diversity than paragraphs used in this work.

It has been observed that paragraphs with many punctuation, numbers and abbreviations were hard to capture by the system. I argue that cleaning the texts and filtering articles with complex linguistic properties might offer data that is better to simplify. Moreover, by focusing on a specific domain the set of challenges might be clearer and easier to tackle since linguistic diversity is reduced.

4.2 Setup

A German version of the medium GPT-2 model GerPT-2 [Minixhofer, 2020] was used for all experiments. First, the model was pre-trained on the *copy task* described in Section 3.2.1.

The training was performed on a computer with a RAM of 64 GB, an I9-9900K processor, and two RTX 2080 Ti GPUs with 11GB memory.

The GUTS training framework consists of multiple transformer models for the generator and reward. These models are distributed on the two GPUs to avoid memory problems. The generator model GerPT-2 and the pre-trained fluency model used in $S_{fluency}$ are employed on the first GPU. The other models for the different scores for the reward were calculated on the second GPU. Moreover, AMP was utilized to save memory during training further and increase the speed. For optimization, AdamW [Loshchilov and Hutter, 2017] was used with a learning rate of $4 \cdot 10^{-5}$. A batch-size of one example was applied in this work and in KiS, meaning after sampling and scoring k simplification candidates conditioned on one original paragraph, the generator is optimized. For experiment tracking and visualization, Weights & Biases has been utilized [Biewald, 2020].

For each original paragraph, $k = 8$ simplification candidates are sampled from the generator. This value was chosen based on the work of Laban et al. [2021b, §4], showing that increasing k results in a higher average reward and a decrease in the reward's variation, boosting the performance and stabilizing the training. In Figure 16, six independent training runs for different k values were conducted, and the average reward was plotted.

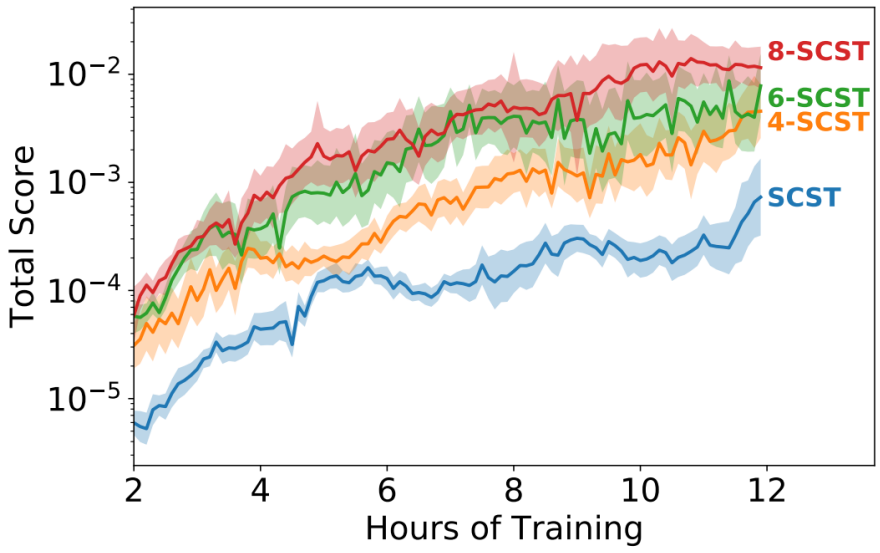


Figure 16: Performances of k-SCST with different values for k [Laban et al., 2021b]

	GUTS-1	GUTS-2
$S_{lexical}$	1.0	0.5
$S_{syntactic}$	2.0	3.0
$S_{meaning}$	3.0	4.0
$S_{fluency}$	0.5	0.5
S_{discr}	0.5	0.5
$S_{brevity}$	1.0	1.0
$S_{hallucination}$	1.0	1.0
S_{ngram}	1.0	1.0
$S_{articles}$	1.0	1.0

Table 5: Score weights used for the training runs

The sampling of the simplification candidates was performed using nucleus sampling with $p = 0.95$, combined with a top-K value of $K = 5$. The eight samples are generated until the end-of-sentence (EOS) token is produced for each one, signalling the end of the generator's output. Following the work of KiS, a setting suppressing the repetition of 5-grams in a sequence was employed during sampling to avoid repeating phrases.

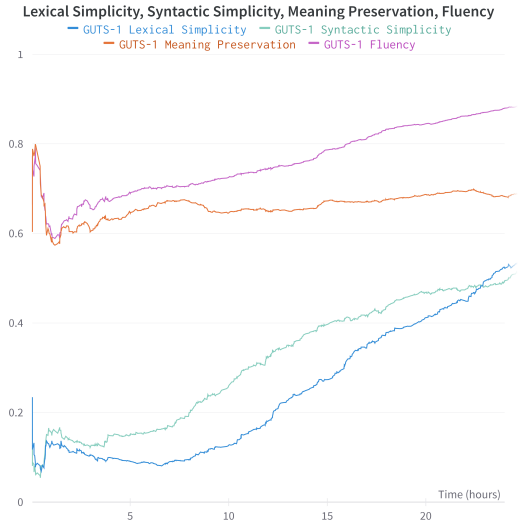
The p value was chosen based on the research of Holtzman et al. [2019]. They argue that values between 0.9 and 1 are the most reliable, and lower values tend to generate repetitions. The $K = 5$ was selected relatively low, as it produced the most reliable results considering the meaning preservation, hallucination and brevity scores in the beginning. In retrospect, the top-K value may have been chosen too low, limiting the diversity of the candidates and restricting the nucleus sampling capabilities.

In the future, better values for nucleus and top-K sampling or even different decoding strategies can be chosen. Within the scope of this work, no extensive testing has been conducted. Additionally, no exact information about the sampling method was described in KiS [Laban et al., 2021b].

It has been found with the experiments that the sampling choice presented stable results in the beginning but lacked diversity in the long run and may have encouraged ungrammatical behaviour in combination with the rewards. Moreover, such a low value for K can be a factor for ungrammatical and non human-like texts since only a small number of probable word candidates are considered. Already outlined in Section 3.2.2, considering only high probability words can degenerate texts and produce repetitions and generic outputs [Holtzman et al., 2019]. Additionally, such a small K value limited the possibilities and improvements that come with nucleus sampling during flat distributions. This aspect can be further tested and improved in future work. Since it produces the basis of the training data, it is essential to sample the best possible candidates for an optimal training signal.

Table 5 shows how the scores during training for the runs of GUTS-1 and GUTS-2 were weighted. This was one of the few differences between the two training runs. The values were oriented on the weights used in KiS, but the meaning preservation score was increased, and the simplicity scores were lowered slightly. Since the scores are adapted and replaced or newly introduced ($S_{articles}$) in this work, it is hard to compare the weights with the approach from KiS.

(a) Mean rewards of GUTS-1 and GUTS-2



(b) Example reward scores from GUTS-1

Figure 17: First 24 hours of training progression

Due to the discrete nature of the guardrail scores, they were clipped to either 0.0001 for failing or 0.9999 for passing. The potential negative impact on the overall reward is therefore higher with these scores ($log(0.0001) = -9.21$) than with the non-guardrail scores, which are clipped between 0.001 and 1 ($log(0.001) = -6.91$).

GUTS-1 was trained for over nine days, GUTS-2 for just about five days. Due to limited time, only a small portion of the training settings and hyperparameters could be explored. There remain many aspects that can be further explored and researched. This will be discussed in detail in Chapter 6. Even though the GUTS-2 ran for only five days, it produced more coherent texts and simplification phenomena than GUTS-1. This may be due to findings that were incorporated during the training of GUTS-1, which will be further explained in the next section.

4.3 Training

Some observations gathered during the main training runs, and some preceding experiments will be described in the following.

Figure 17 shows two plots visualizing the training progression. The lines plotted are smoothed using an exponential moving average over the data points. Figure 17a shows GUTS-1 and GUTS-2 mean rewards for the eight sampled candidates at each training step. The mean reward shows an upwards trend in this plot. Scores for lexical and syntactic simplicity, meaning preservation and fluency are visualized in Figure 17b, which are also averaged values of the eight sampled candidates for each training step. All scores, except meaning preservation, are increasing. $S_{meaning}$ starts out relatively high since the generator was already pre-trained to copy the original paragraph (explained in Section 3.2).

GUTS-2's mean reward stagnated after two days and did not further increase. This might be optimized by adjusting training parameters like the learning rate and by increas-

ing the diversity of the samples.

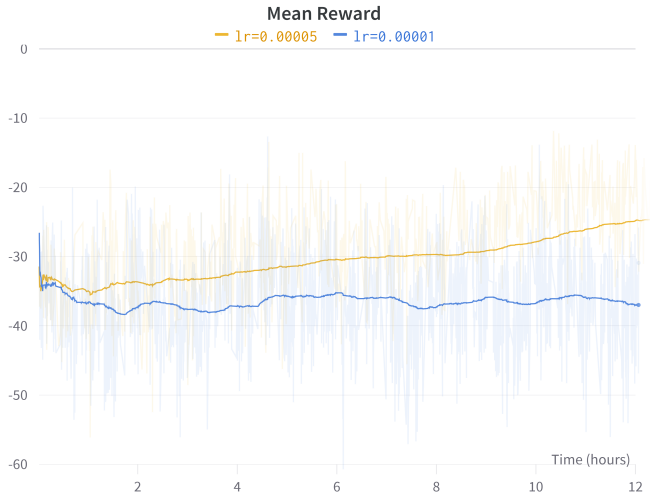


Figure 18: Reward optimization with different learning rates

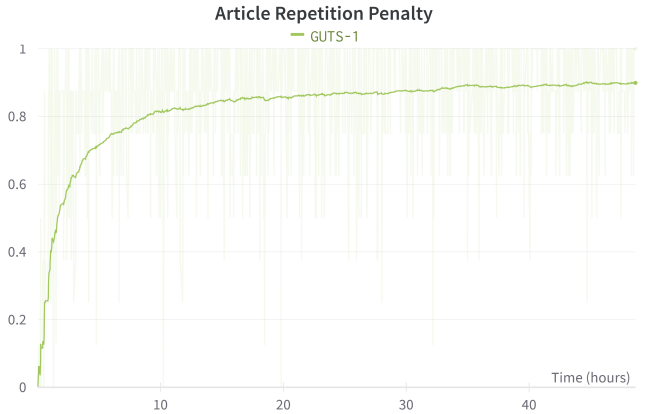


Figure 19: Plot of $S_{article}$ after introducing it to GUTS-1

Before settling for a learning rate of $4 \cdot 10^{-5}$ for the two main training runs, the first experiments started out with a value of 10^{-6} following the choice by Laban et al. [2021b, §A.1]. Using such a low learning rate the reward and scores during these experiments did not increase as expected. Next, an even lower learning rate (10^{-7}) has been tried, but this also did not increase the training performance. Figure 18 shows a comparison between two runs for 12 hours. One run with a learning rate of $5 \cdot 10^{-5}$ has an upwards trend in the reward (yellow line), while the blue line represents the run with a learning rate of 10^{-5} stagnates. Due to time constraints, no further testing on the learning rate was performed. Extensively experimenting with different learning rates in the future would be insightful for this approach.

Optimizing the learning rate during the experiments, while subsequently tuning and experimenting with the reward scores and other training hyperparameters, made finding a more optimal learning rate difficult. The reward and the function and weights of its

individual scores directly influence the loss, after which the learning rate must be adjusted accordingly. An adaptive learning rate would be an interesting addition to try out with k-SCST. Additionally, a decaying learning rate could be incorporated like Rennie et al. [2017] used for their SCST approach.

Already highlighted in Section 3.1.3, some problems with the fluency model have been noticed. The LM-fluency score $S_{fluency}$ often assigned high scores to samples that contained grammatical errors, which will be further evaluated in Section 5.2. Figure 10 showed an example where the generator learnt to cheat the scores by producing repeating articles, like "der"/"die"/"das"/... (English: "the"). The LM fluency model score often highly rewarded such samples. By introducing the Article Repetition Penalty score, this behaviour was penalized, and the generator seemed to unlearn these patterns. This cheating of the generator model was noticed after roughly three days of training. After incorporating $S_{article}$ and continuing the training, this repeating behaviour was quickly averted, as shown in Figure 19. Despite adapting to this score, GUTS-1 still had learnt some fundamental grammatical flaws in these first three days. It showed significantly more fluency mistakes than GUTS-2 during evaluation, although it was trained for longer.

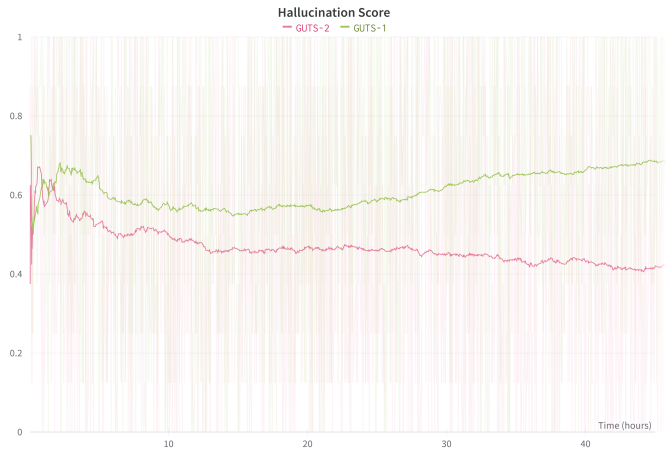


Figure 20: Plot of $S_{hallucination}$ for GUTS-1 and GUTS-2

Another noticeable difference between the training runs of GUTS-1 and GUTS-2, was the score $S_{hallucination}$. The first 45 hours of this score for both runs are visualized in Figure 20. GUTS-2 did not improve this score during training. It started with an average of 0.5 - showing that around half of the produced simplification candidates contained hallucinations according to the score. This value only decreased to slightly above 0.4. GUTS-1, on the other hand, learned to increase this score to an average of around 0.8 by the end of its training time.

Figure 21 shows an example with three samples and their respective reward scores that have been generated based on the original paragraph. These particular examples are from GUTS-2's training run. During training the original paragraph, the samples and corresponding rewards have been logged every 150 training steps. This helped to find problems during the experiments.

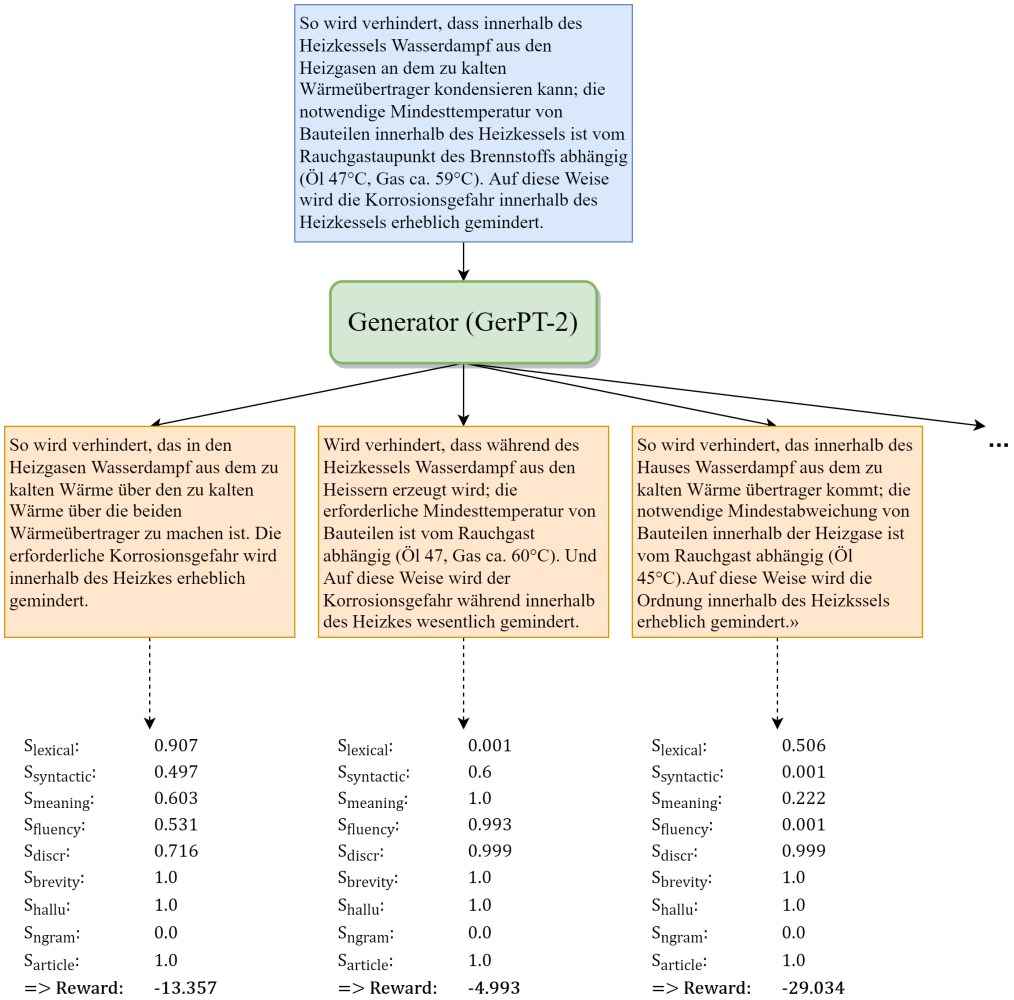


Figure 21: Example samples with rewards from training GUTS-2

Chapter 5

Evaluation

In this chapter, the models *GUTS-1* and *GUTS-2* are evaluated. First, the procedure for the automatic evaluation is explained. Here the setup of the data, the evaluation metrics and the models are outlined. For the manual evaluation, a qualitative analysis of some examples is performed, where some simplification phenomena and common problems with GUTS are assessed.

5.1 Automatic Evaluation

For the automatic evaluation, the procedure of KiS has been tried to be followed [Laban et al., 2021b, §5.2]. Since there exist no comparable German models that can simplify on a paragraph-level, a Pivot model is introduced, based on the released version of KiS and two translation models. This model will be used as a baseline for the automatic evaluation as it can simplify whole paragraphs [Laban et al., 2021b].

5.1.1 Data

Since there are no qualitative German simplification datasets with whole paragraphs, a dataset for evaluation based on TextComplexityDE was manually assembled. TextComplexityDE consists of multiple sentence-pairs with simplifications from different articles. Several paragraphs were constructed by concatenating individual sentences. Because these sentences often were not contiguous, the paragraphs were carefully composed. On the one hand, the combined sentences must fit sequentially as logical as possible. On the other hand, the paragraphs were created to be within the range that the GUTS models were trained on (80-175 tokens). Table 6 shows two examples paragraphs from the evaluation dataset.

Besides the TextComplexityDE dataset, 300 paragraphs from the training dataset of Wikipedia articles have been randomly selected. This dataset is composed of texts with different lengths and topics. The length has been chosen according to the three length categories described in Section 4.1. From each category, 100 paragraphs have been selected to have a balanced distribution of different length texts. This dataset contains articles that are linguistically more diverse and difficult than those from the TextComplexityDE dataset, which consists of manually picked sentences from Wikipedia articles and their simplifications.

Original	**Simplification**
Neben den Füßen wird beim Radfahren auch das Gesäß belastet. Das menschliche Becken besitzt Sitzbeinhöcker und Schambein bzw. Schambeinkufen (auch Schambeinkamm genannt), die beim Fahrradfahren gekippt werden. Dieser Kippwinkel hat Auswirkungen auf die Biegung der gesamten Wirbelsäule. Deshalb kann dieser Berührungspunkt problematisch sein.	Als zweitgrößte Belastungszone ist der Kontakt zwischen Gesäß und Sattel anzusehen. Dieser Berührungspunkt kann insbesondere daher problematisch werden, da das menschliche Becken mit seinen Sitzbeinhöckern und dem Schambein (bzw. den Schambeinkufen, auch Schambeinkamm genannt) durch den Kippwinkel des Beckens Auswirkungen auf die Biegung der gesamten Wirbelsäule hat.
Dabei können die Ergebnisse addiert werden (eine Waffe in einem Rollenspiel richtet soviel Schaden an, wie zwei Würfel zusammen anzeigen) oder als Ensemble betrachtet werden (bei vielen Brettspielen folgen besondere Aktionen, wenn mehrere Würfel die gleiche Zahl zeigen, bei einem sogenannten Pasch). Um das Werfen mehrerer Würfel zu vereinfachen, Schummeln durch Trickwürfe zu vermeiden oder das Ergebnis vor anderen Spielern zu verbergen, kommen Würfelbecher (Knobelbecher genannt) zum Einsatz.	Dabei können Ergebnisse addiert oder zusammengehörendend betrachtet werden. Eine Waffe richtet dabei in einem Rollenspiel soviel Schaden an, wie zwei Würfel zusammen anzeigen. Wenn mehrere Würfel die gleiche Zahl anzeigen (sogenannter Pasch) folgen bei vielen Brettspielen besondere Aktionen. Damit man einfacher mehrere Würfel werfen kann, werden Würfelbecher (Knobelbecher) benutzt. Außerdem vermeidet man Schummeln und kann das Ergebnis vor anderen Spielern verbergen.

Table 6: Examples from the TextComplexityDE evaluation data.

5.1.2 Metrics

Since there is no single agreed-upon measurement for simplicity [Alva-Manchego et al., 2021], a combination of reference-based and reference-less metrics has been used.

SARI: Previously described in Section 2.1.3, SARI was integrated as a reference-based simplification metric. SARI showed the best correlation with human judgements on simplicity gain compared to other automatic metrics. [Alva-Manchego et al., 2021] The metric was originally designed to operate on a sentence-level [Xu et al., 2016], but also has a corpus-level version implementation. However, the authors only tested the correlation with human judgements of SARI at a sentence-level.

FRE: Additionally to SARI, the syntactic simplicity is measured with the FRE, which has been also used as the basis for the syntactic simplicity score. For evaluation, the mean FRE of the models' outputs *FRE(S)* and the average difference between the FRE value of the original text and the simplification, referred to as *FRE diff*, are calculated. If the difference is > 0, the paragraph's syntactic simplicity has increased during the simplification process.

Zipf: The average Zipf values of all non-stop words are used to measure the lexical simplicity improvement *Zipf diff* between the original paragraph P and the simplification S by taking the difference: $\overline{Zipf}(S) - \overline{Zipf}(P)$. Again, if the result is > 0, the simplification uses more frequent words and arguably simpler ones [Breland, 1996].

Meaning Preservation: The score described in Section 3.1.2 $S_{meaning}$ is used to capture the meaning adequacy of the simplifications. With this score, the models are rated on how well the contents from the original paragraph are preserved.

Compression: Lastly, the compression rate (*Comp.*) is measured by dividing the length of the system's simplification by the original paragraph: $\frac{length(S)}{length(P)}$

Laban et al. [2021b, §5.2] furthermore integrated the BLEU metric. BLEU, short for bilingual evaluation understudy, was originally developed for machine translation [Papineni et al., 2002] but has since been included in the evaluation of ATS systems. In this work, BLEU has been excluded from the evaluation. Sulem et al. [2018a] found BLEU unsuitable for text simplification since it penalizes operations like sentence splitting, a common simplification phenomenon.

5.1.3 Models

For the automatic evaluation, the two GUTS models, described in Section 4, are used. Since there were no comparable German models available that can simplify on a paragraph-level, a *Pivot model* is introduced for evaluation, consisting of two machine translation models and one simplification model. This Pivot model is inspired by a similar model introduced by Mallinson et al. [2020], which they used as a comparison in their evaluation. The functionality with an example text is presented in Figure 22. First, the paragraph is translated from German to English by the one translation model (de-en). The KiS model can then simplify the English paragraph. The second translation model (en-de) then translates this simplification back to German. The resulting simplification is used

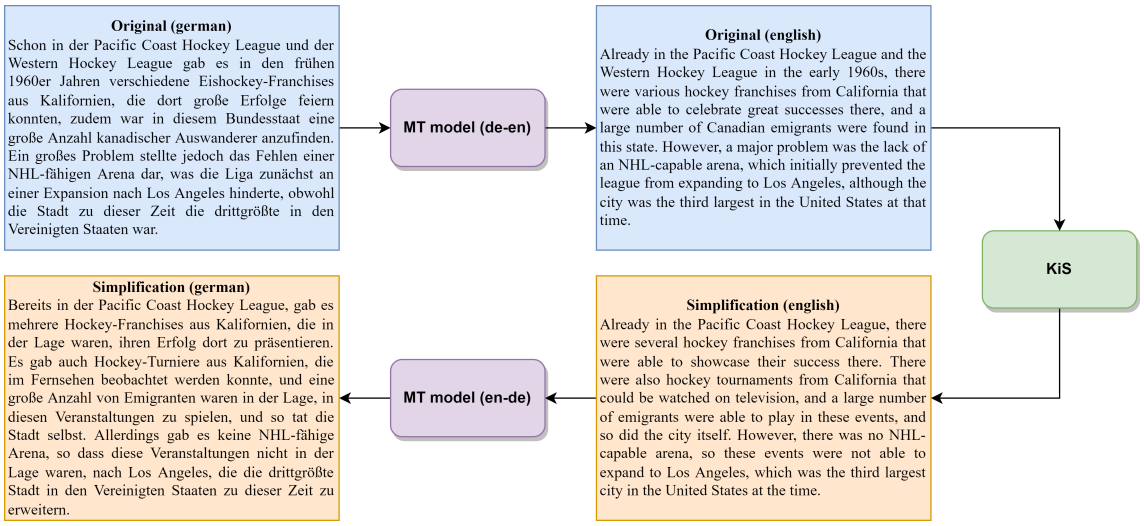


Figure 22: Simplification example of the Pivot model

for evaluation. The simplification model is based on a released version of KiS [1]. The KiS model, a medium GPT-2 model, was trained on a large corpus of English news articles [Laban et al., 2021b, §A.1].

For the translation models, two trained transformers have been used [Tiedemann and Thottingal, 2020]. The first model[2] was trained for translating German to English and is used to translate the original paragraphs from the evaluation dataset. The English simplification output by the KiS model is then re-translated into German by the second translation model[3]. Both of these translation models consist of a transformer architecture, where neural MT training is performed. Additionally, an alignment loss is constructed to learn word alignments between two languages. [Garg et al., 2019]

The Pivot model offers a good baseline to compare the GUTS simplification model against since it is trained in the same way and uses similar metrics and methods to reward the candidate samples. A further example simplification can be found in A.1.

All simplifications were received from the models using greedy decoding for the automatic evaluation. Other sampling methods have been tested, but the values for the previously described metrics did not show a significant difference. The greedy results were therefore kept for the results.

5.1.4 Results

Table 7 displays the automatic results on the adapted TextComplexityDE dataset. GUTS-1 and GUTS-2 are slightly outperformed by the Pivot model on SARI. All models improve on the FRE difference and with that an arguably syntactic simplification. GUTS-1 shows the most significant improvement out of the three models but is outperformed by the human-written reference simplification. All models show a positive Zipf difference. GUTS-1 outperforms the other models, while GUTS-2 shows almost no improvement on this met-

[1] https://github.com/tingofurro/keep_it_simple/releases/tag/0.1
[2] https://huggingface.co/Helsinki-NLP/opus-mt-de-en
[3] https://huggingface.co/Helsinki-NLP/opus-mt-en-de

Model	SARI	FRE(S)	FRE diff	Zipf diff	Meaning	Comp.
TcDE	-	46.847	21.194	0.274	0.896	0.933
GUTS-1	0.348	41.361	15.71	0.231	0.937	0.803
GUTS-2	0.348	37.448	11.795	0.059	0.875	0.789
Pivot	0.370	38.712	13.059	0.206	0.727	0.863

Table 7: Automatic results of TextComplexityDE

Model	FRE(S)	FRE diff	Zipf diff	Meaning	Comp.
GUTS-1	56.198	12.413	0.278	0.748	0.735
GUTS-2	53.130	9.376	-0.001	0.819	0.731
Pivot	50.187	6.402	0.243	0.549	0.766

Table 8: Automatic results of Wikipedia paragraphs

ric. This difference between the GUTS models can be explained by the shorter training time of GUTS-2, and lower weight for the lexical simplicity score $S_{lexical}$. All models performed reasonably well regarding the meaning preservation. GUTS-1 even outperformed the reference since it was directly trained on this score. The Pivot model lacks behind in this area, indicating that its simplifications did not capture as much information from the original paragraph, according to $S_{meaning}$. All models tend to shorten the texts during simplification, shown by the similar compression values. The reference simplifications stay closest to the original paragraph's length.

In Table 8, the evaluation on the Wikipedia paragraphs, using only the reference-less metrics, is presented. Again GUTS-1 shows the best improvements regarding FRE and Zipf. While GUTS-2 achieves better FRE values than the pivot model on this data, it repeats to have the worst results on the Zipf improvements, showing no gain on this metric. GUTS-2 achieves the best meaning preservation scores on this dataset, closely followed by GUTS-1. The Pivot model performs worse than both GUTS models for this aspect. For the compression rate, all models show comparable values regarding their simplifications lengths.

These automatic results can only capture the simplicity objective to a certain degree. To further evaluate the performance of GUTS, a manual evaluation has been conducted, which is presented in the next section.

5.2 Manual Evaluation

Besides automatic metrics, ATS systems are commonly evaluated using a study, relying on human judgement on how good the simplifications of different systems are. Often, these are rated using a five-point Likert scale for three criteria (grammaticality, meaning preservation/adequacy, and simplicity). [Mallinson et al., 2020; Zhang and Lapata, 2017; Martin et al., 2019]

With KiS, Laban et al. [2021b, §5.3] proposed a novel manual evaluation to assess the usefulness of simplification results. The study is based on the assumption that simplified

texts should be faster to read and comprehend. First, they took the original texts and reference simplifications from the Newsela dataset. Secondly, they used different models for the evaluation to generate simplifications for the original Newsela articles. For each of the documents, they selected five multiple-choice questions that have been generated by a GPT-2 model, fine-tuned on question generation. The participants' task was to read a random article and answer the questions until they submitted them correctly. They measured the number of submissions until the questions were answered correctly and the question completion time in seconds. The KiS model's simplifications led to a significant speed-up compared to the other texts.

An evaluation like this was not performed for this work since the simplifications generated by the GUTS models showed problems with faithfulness and fluency. There are various aspects, where this approach can be improved. The evaluation with a human study is left for future work.

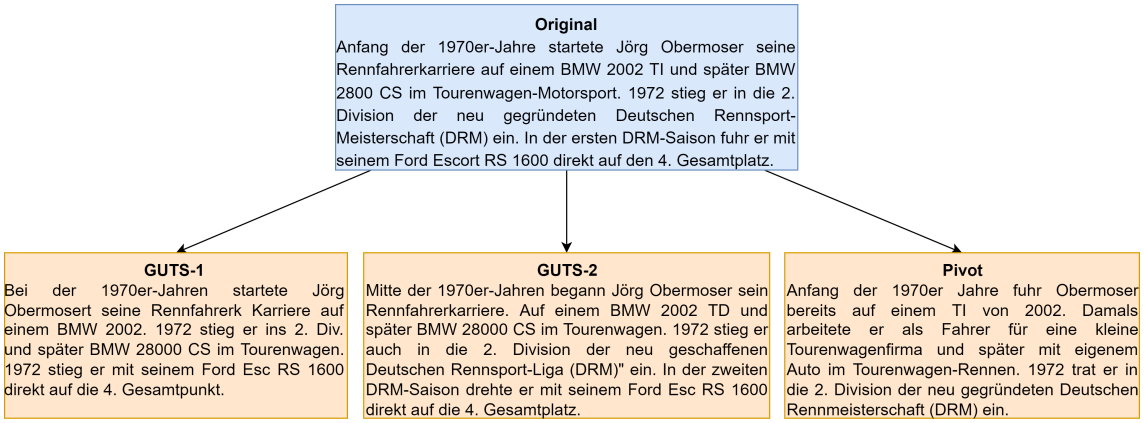


Figure 23: Simplifications examples from the evaluated models

In the following, the simplifications produced by the models are manually evaluated. Note that these are observations by the author, focusing on simplification phenomena and common problems with GUTS. This is done to guide future work to improve the system. The evaluation is primarily performed for the outputs of GUTS-2 since GUTS-1 almost always produced simplifications with grammatical mistakes, sometimes even resulting in incoherent language. While the GUTS-2 model also regularly contained non-fluent parts, the texts were much more coherent. GUTS-1 and the Pivot model are only used for specific examples or comparisons. Figure 23 shows a paragraph from a Wikipedia article used for evaluation and the generated simplifications by the models. The texts in this figure were all produced using greedy decoding. In this example, the simplification by GUTS-1 shows multiple grammatical mistakes. GUTS-2 also contains a small error but is more coherent.

For this evaluation, the 300 Wikipedia paragraphs, that have been also used in the automatic evaluation, are reviewed. The GUTS-2 outputs were sampled using a combination nucleus sampling with $p = 0.95$, combined with a top-K value of 100 and a temperature of 0.6 to increase the model's confidence in already high probable words. It was also experimented with other decoding methods, but this setting offered good simplification results.

First, some simplification phenomena are listed that could be observed with the system's output. The overall quality of these simplification occurrences is assessed. In the

subsequent section, common problems discovered during the evaluation are stated, as well as some possible causes for these mistakes.

5.2.1 Simplification phenomena

Lexical simplifications

Original	Simplification
Für sein literarisches **Werk** wurde Shimao **vielfach** ausgezeichnet.	Für sein literarisches **Buch** wurde Shimao **mehrfach** ausgezeichnet.
Eine Abstammung aus dem Geschlecht der Earls of Loudoun aus dem Clan Campbell, die aus derselben Gegend entstammten, ist trotz des gleichen Wappens nicht **belegt.**	Eine Abstammung aus der Geschlecht der Earlen von Loudoun, die aus derselben Region entstammen, ist trotz des gleich Wappens nicht **bestätigt**.
Da er mit den in der **pomologischen** Literatur veröffentlichten Sortenbeschreibungen **Probleme** hatte, seine eigenen **Sorten** zu identifizieren, legte er eigene **detaillierte** Beschreibungen und Schnittzeichnungen an.	Der in der **pomographischen** Literatur veröffentlichten Sortenbeschreibungen **Schwierigkeiten** hatte, seine eigenen **Pflanzen** zu erkennen, legte er eigene **genaue** Beschreibungen und Schnittzeichnungen.

Figure 24: Excerpts of lexical substitutions by GUTS-2

With GUTS-2, some lexical simplifications in the form of substitutions with synonyms could be observed. Most of the examples were not necessarily simpler. Examples are shown in Figure 24. All bold words are lexical substitutions, the black coloured words signal substitutions that did not make the text really easier, but were nonetheless analogous to the original word. These occurred most dominantly with the studied simplifications. The *Zipf diff* values in Table 7 and 8 already indicated that lexical simplicity improvements, measured by word frequency, were almost non-existent with GUTS-2. The last example in Figure 24 shows a faulty word substitution marked in red. Mistakes like this were sometimes observed and made the text unfaithful since the word "pomographisch" does not exist in German. The substitutions in green indicate a simplification. By generating the word "Pflanzen" (English: "plants"), the model showed that it has some context about the topic. While "Pflanzen" and "Sorten" (English: "kinds") are no direct synonyms, in this example the substitution is not faulty and does not change the content of the original paragraph. Lexical simplifications of this kind were observed rarely with the generated data. A longer training time and an increased weight for the lexical score might increase the quantity and quality of the lexical substitutions.

Instead of substituting with synonyms, many lexical changes in the simplifications were word length reductions generated by GUTS-2 and GUTS-1. With this, a part of a composed word was deleted during simplification and the other part was kept. These did sometimes result in arguably easier words without changing the content of the text. In Figure 25, words that got changed by reducing the length are marked bold. Again "good" examples are highlighted in green, "bad" examples in red. A good example is found in the first excerpt: the word "Schlossräume" (English: "palace rooms") has a Zipf value of 1.08 and the substitution "Räume" (English: "rooms") a value of 4.4, indicating a lexical simplification without changing the semantic of the sentence too much. Shortening of this kind can make a text easier to understand since word length has also been shown to

Original	Simplification
Erst ab 1860 unter Fürst Adolf Georg wurden die **Schlossräume** renoviert;	Erst ab 1860 unter Fürsten Adolf Georg wurden die **Räume** renoviert;
Sie hatte Marschall Tito als Jugoslawiens **Präsidentenflugzeug** gedient und fand in den 1970er Jahren auch in Sambia bei dessen Präsidenten Kenneth Kaunda Verwendung.	Sie hatte Marschalls Tito als Jugoslawien **Präsident** gedient und fand in die 1970er Jahre auch in Sambia.
Im 16. Jahrhundert entwickelte sich über die **Divinationen** eine Vielzahl an **Schriftgut**, schließlich führte dieses Wissen über die alten Vorstellungen zu einer Zusammenfassung und entwickelte ein eigenes **Wissensgebiet**.	Im 16. Jahrhundert entwickelt sich über die **Divinen** eine Vielzahl an **Schrift Gut**, schließlich führte dieses wissen über die alten Vorstellungen und entwickelte ein eigenes **Wissen**.

Figure 25: Excerpts of word reductions by GUTS-2

correlate with complexity [Keskisärkkä, 2012].

The second example in Figure 25 shows a faulty word reduction. The word "Präsidentenflugzeug" (English: "presidential plane") was reduced to only "Präsident" (English: "president"), which removes important information from the sentence and gives the reader a misleading simplification.

The last example in the figure shows multiple word reductions: With the first word reduction, GUTS-2 generated a word that does not exist in German ("Divinen"). The second instance in this example, split up the word "Schriftgut" (English: "Document") into "Schrift" (English: "Script") and "Gut" (English: "Property"). Both of these words exist in German, but this split makes the sentence non-fluent and may be confusing for the reader. However, according to $S_{lexical}$, this made the text simpler since "Schriftgut" has a Zipf value of 2.3, while the individual words "Schrift" a value of 4.32 and "Gut" a value of 6.09.

These word length reductions are encouraged by the lexical score $S_{lexical}$. The individual parts of a word composition are generally used more frequently than the composition. Additionally, this behaviour can also be encouraged by $S_{syntactic}$ since shorter words in a sentence improve the score. Moreover, word reductions with names were regularly discovered with GUTS-2, which made the texts unfaithful. This is further explained in Section 5.2.2.

Structural changes

First, we look at the common simplification phenomenon of sentence splitting: Sentence splitting in text simplification is when a long sentence is split up into two or more short sentences, each carrying less information than the original sentence. Optimally every sentence should carry one statement to be easily understandable [Maaß, 2015]. This kind of operation was infrequently found among the simplifications of GUTS.

In Figure 26, three sentence splitting examples that occurred with the simplification of GUTS-2 have been observed. The first two examples seem relatively comprehensible and split the information of one sentence up into two sentences. In the third excerpt, GUTS just replaced a comma with a period, making the text arguably less fluent and harder to comprehend. This behaviour is encouraged by the $S_{syntactic}$ since the average sentence length of this text decreased.

Original	Simplification
Anfang der 1970er-Jahre startete Jörg Obermoser seine Rennfahrerkarriere auf einem BMW 2002 TI und später BMW 2800 CS im Tourenwagen-Motorsport.	Mitte der 1970er-Jahren begann Jörg Obermoser sein Rennfahrerkarriere. Auf einem BMW 2002 TD und später BMW 28000 CS im Tourenwagen.
Unter Mitarbeit von Ulrike Kiefer, Mitherausgeberin des "Language and Culture Atlas of Ashkenazic Jewry," begann das EYDES-Projekt an der Ruhr-Universität Bochum in den Jahren 1996 bis 2000 mit einem umfangreichen Vorhaben. Die Transkription, Digitalisierung und Erschließung des Materials des LCAAJ durch ein webbasiertes Datenbankinterface wurde durchgeführt.	Bei Mitarbeit von Ulrike Kiefer" war das EYDes-Projekt an der Universität Bochum in den Jahren 1998 bis 2000. Mit einem umfangreichen Vorhaben. Das Transkript, Digitalisierung und Erschließung der Materialien des LCAA durch ein webbasierte Datenbankinterface wurde ausgeführt.
Obwohl der unmittelbare japanische Einfluss 1945 endete, führten die Wirtschaftsaktivitäten in der Region zu einer Mentalitätsänderung, die für viele asiatische Länder nach 1945 von signifikanter Bedeutung für die weitere wirtschaftliche Entwicklung war.	Obwohl der direkte japanische Einfluss 1945 endete. führten die Wirtschaftsaktivitäten auch in der Region zu eine Mentalitätsänderung, das für viele asiatische Länder von beachtlicher Bedeutung für die weitere ökonomische Entwicklung war.

Figure 26: Sentence splittings generated by GUTS-2

Even though sentence splitting is one of the most common syntactic simplification operations, the sampled candidates combined with the reward were not a good enough training signal for GUTS to learn consistent sentence splitting. This has to be confirmed with more experiments and a longer training time. Similar behaviour has been observed with the Pivot model. Both models tend to delete parts of the text to make them shorter rather than splitting them. Deletions can be observed with most examples used in this evaluation and is encouraged by $S_{syntactic}$.

Original	Simplification
Die Rasenfläche ist an beiden Seiten von vorzugsweise heimischen Gehölzen begrenzt. Insgesamt befinden sind rund 30 Baumarten im Park, darunter Linden, Eichen, Erlen, Buchen, Ulmen, Eschen und der Ahorn, teilweise auch einzelne Fichten und Tannen.	Die Rasenfläche ist am beiden Seiten von vorzugsweise einheimischen Gehölzen begrenzt, wobei rund 30 Baumarten in Park, darunter Linden und der Ahorn stehen.
Darüber hinaus wurde er zum Ehrenmitglied der Gesellschaft für Geologische Wissenschaften der DDR, der Tschechoslowakischen Gesellschaft für Mineralogie und Geologie sowie der Ungarischen Geologischen Gesellschaft ernannt.	Darüber hinaus wurde er zum Ehrenmitglied des Vereins für Geologische Wissenschaften gewählt.

Figure 27: Deletions performed by GUTS-2

Figure 27 shows two examples where parts of the text have been deleted by GUTS-2. In this case, arguably the most important statement of the sentence is preserved. The first example just shortens an enumeration of tree species. This way the text is shorter and easier to comprehend. In the second example, large parts of the original sentence are removed during simplification. Arguably the most important statement is preserved, making the sentence faster to read and easier to comprehend.

Deletions can help the reader understand texts better by removing non-essential information that may be confusing to a low literacy reader. Specifying what parts of the text are important and what can be removed is a context-dependent task and can not be generalized. The original sentence structures in both examples were rather complex

and nested, GUTS-2 showed it can reduce the complexity by removing the rather less important parts of the text. Besides, shorter sentences are easier to understand for people with aphasia and autism [Carroll et al., 1999; Evans et al., 2014].

5.2.2 Problems

Grammatical Mistakes

Guaranteeing the fluency and readability of a text is one of the most critical aspects of natural language generation tasks such as text simplification. One big problem that mitigates the quality of simplifications produced by GUTS is ungrammaticality. In most of the evaluated simplifications, grammatical errors could be observed. Many of these were minor errors, like confusing German articles, e.g. using "das" instead of "der" or making mistakes with the tense of a word, for example, using the present instead of past tense.

Original	**Simplification**
Das Egelsee-Ried wurde am 12. Juli 1983 zum Naturschutzgebiet erklärt. Das 5,29 Hektar große Naturschutzgebiet ist vollständig in Landesbesitz. Es besteht aus einem Niedermoor und einer von 1972 bis 1982 betriebenen und inzwischen stillgelegten Kiesgrube.	Das Egelssee-Ried war am 12. Am 12. Am 17. Am 17." ist komplett in Landesbesitz. An besteht aus einem Niedermoen und einer von 1972 über Betrieb und inzwischen stillgelegten Schotter.
Die beiden Maschinen gingen 1992 bzw. 1994 an Namibia Commercial Aviation, wurden restauriert und seitdem für Charterflugreisen innerhalb Namibias eingesetzt.	Die beiden Maschinen gehen 1992 bzw. 1994 für Charterflüge innerhalb Namibias eingesetzt.
Der Ausbau der Infrastruktur begann mit dem Bau eines Post- und Telegraphenamtes im Jahr 1873 beziehungsweise 1898 mit der Errichtung eines Bahnhofs an der Donauuferbahn und wurde unterbrochen durch die Kriegsjahre im Einklang mit der Entwicklung im ganzen Land fortgeführt.	Der Ausbau der Struktur begann mit dem Bau einer Post- und Telegrafamt im Jahr 1873 respektive 1898 mit der Errichtung einer Bahnhof an der Donauufer und wurde unterbrochen durch den Kriegsjahre im Einklang zwischen der Entwicklung im ganzen Lande fortgeführt.

Figure 28: Grammatical errors produced by GUTS-2

As already stated in Section 4.3, the simplifications by GUTS-1 almost always contained major grammatical mistakes and sometimes completely incomprehensible outputs. GUTS-2 regularly produced some of the previously mentioned grammatical issues but rarely had completely incoherent outputs like GUTS-1. The first example in Figure 28 shows an incoherent text produced by GUTS-2. In this excerpt, multiple mistakes are present, like word repetitions and other grammatical errors that lead to a text that is hard to understand. When rating this simplification with $S_{fluency}$, a score of 0.994 is returned. This suggests that the method of scoring the fluency of a generated text needs improvement.

The second example in Figure 28 shows a non-fluent text caused by removing parts of the original sentence. The rest was closely copied, resulting in a confusing and ungrammatical sentence. The third example shows some small grammatical errors caused by shortening original words and using the wrong articles for some nouns.

The Pivot model showed the least amount of grammatical mistakes. The produced simplifications that were observed in this were almost always fluent.

Original	Simplification
Oskar Gustav Rudolf Hergt, auch "Oscar" (* 22. Oktober 1869 in Naumburg; † 9. Mai 1967 in Göttingen), war ein deutscher Verwaltungsjurist und Politiker der Deutschnationalen Volkspartei (DNVP). Er war von 1918 bis 1924 Parteivorsitzender der DNVP, von 1920 bis 1933 Mitglied des Reichstages sowie von 1927 bis 1928 Reichsjustizminister und stellvertretender Reichskanzler im Kabinett Marx IV.	Oskar Gustav Hergt, auch von "Oscar" (18. Oktober 1869, in Naumburg; † 9. Mai 1967) in Göttingen, war ein deutscher Verwaltungsrechtler. Er war von 1918 über 1922 Parteivorsitzender der DnVP, von 1920 über 1924 Mitglied des Reichstages und von 1928 Reichsjustizminister.
Blutbrust-Bartvögel erreichen eine Körperlänge von etwa 15 Zentimeter. Die Männchen der Nominatform haben eine Flügellänge von 7,7 bis 8,7 Zentimeter. Der Schwanz wird zwischen 4,2 und 4,7 Zentimeter lang. Die Schnabellänge beträgt 1,8 bis 2,1 Zentimeter. Weibchen haben ähnliche Körpermaße. Es besteht kein auffälliger Sexualdimorphismus.	Bartvögel erreichen bis eine Körperlänge von ungefähr 10 Zentimetern. Die Männchen der nominatform haben eine Fliegenlänge von 7,5 bis 8,8 Zentimeter. Der Schwanz ist zwischen 4,2 bis 4,7 Zentimeter. Die Schnabellose beträgt 2,8 bis 2 Zentimeter. Die Schnabel beträgt 1,8. Weibchen haben ähnliche Form. Es besteht kein Auffälliger Sexualdim.

Figure 29: Date and numeric mistakes produced by GUTS-2

Faithfulness

Another common issue the models had, were problems with faithfulness, already briefly mentioned in Section 3.1.4. Factual inconsistencies between source and generated texts were frequently observed with the simplifications of GUTS-2. One of the most common inconsistencies was numeric values, such as dates or measurements. This often happened when multiple different dates and numbers were present in an original paragraph. Figure 29 shows two examples where mistakes of this nature were generated by GUTS-2 (highlighted in red). In the first example, multiple dates are contained in the original text. In the simplification, these are mixed up or entirely hallucinated. The second excerpt contains measurements that GUTS-2 incorrectly adopted. These inconsistencies with numbers were not considered by any scoring method for the reward. For future work, the score for hallucination detection $S_{hallucination}$ could be extended to take numbers and dates into account, like Laban et al. [2021b] did in their approach.

Besides these generated mistakes, other faithfulness issues, like hallucinations, were found with the simplifications of GUTS-2.

Original	Simplification
Nach dem Wahlsieg der DLP bei den Wahlen vom 15. Januar 2008 wurde er Botschafter bei der Karibischen Gemeinschaft (CARICOM), ehe er am 23. Oktober 2010 Parlamentarischer Staatssekretär im Ministerium für Umwelt, Wasserressourcen und Entwässerung wurde.	Nach dem Wahlsieg bei den Wahlen vom 16. Januar 2008 wurde er Botschafter bei der Amerischen Gemeinschaft (CARD), ehe er am 22. Oktober 2010 Parlamentlicher Staatssekretär im Ministerium wurde.
Erst ab 1860 unter Fürst Adolf Georg wurden die Schlossräume renoviert;	Schon ab 1860 unter Fürst Ad wurden die Schlossräume für den Bau und die übrigen Zimmer renoviert.
Oskar Gustav Rudolf Hergt, auch "Oscar" (* 22. Oktober 1869 in Naumburg; † 9. Mai 1967 in Göttingen), war ein deutscher Verwaltungsjurist und Politiker der Deutschnationalen Volkspartei (DNVP).	König Ludwig I. (* 19. Oktober 1866 in Naumburg; 18. Mai 1967 in Göttvard), war ein deutscher Regierungsjurist und Politiker.

Figure 30: Hallucinations generated by GUTS-2

In Figure 30, the first example shows a hallucination by GUTS-2, where a name and

abbreviation were wrongly adopted by the model. In the second instance, the model produced such a mistake by shortening the name. This common behaviour was touched on in Section 5.2.1 with the described word length reductions. The third example shows a hallucination where the original name is replaced by the entirely unrelated entity "König Ludwig I.". Hallucinations like the third were the least common ones with the observed simplifications.

Surprisingly GUTS-1, which generated overall the worst simplifications among the models, generated less hallucination with names. Already shown in Section 4.3 during the training of GUTS-2, the hallucination score did not improve, while GUTS-1 learned to eliminate the hallucination of new entities.

The results of the pivot model rarely contained the previously mentioned faithfulness issues. I hypothesize that this has two reasons: First, as already stated the Hallucination Detection score (Section 3.1.4) did not increase with GUTS-2 as expected, which might change with different training settings and longer training time. Secondly, the coverage model in KiS to measure the meaning preservation might penalize the usage of wrong keywords such as names and locations, additionally to the hallucination detection [Laban et al., 2021b, §3.3]. More training time and experimenting with the settings and hyperparameters are needed to confirm these assumptions.

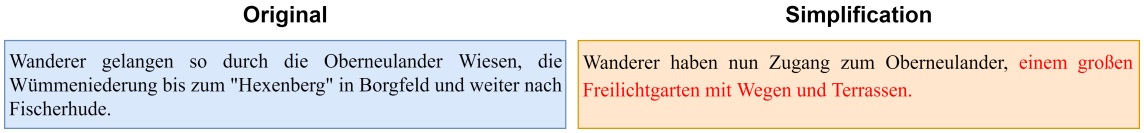


Figure 31: Faithfulness error generated by the Pivot model

The pivot model has been found to introduce new sentences or phrases to the simplification, that were hallucinated. An example is given in Figure 31. The GUTS models rarely generated such mistakes. I assume that this again is because of the different measurements for the meaning preservation. While KiS's coverage model tries to fill in the masked keywords, it does not directly assess if unnecessary content has been added to the simplification. The score $S_{meaning}$ penalizes such behaviour.

Chapter 6

Discussion

This thesis described the technical implementation, training and evaluation of GUTS, an unsupervised simplification approach. This approach is an adaption of KiS by Laban et al. [2021b] for the German language. All parts of the original system KiS have been revisited, some of which needed to be adjusted or redone for this work. This chapter discusses the presented approach. The following sections summarise the contributions, limitations and ideas for future work.

6.1 Contributions

There is still a data scarcity for German simplification datasets. So far, the available datasets are not large enough to sufficiently train a neural model in a supervised way. This work showed a way to bypass this data shortage by approaching the problem with the RL learning algorithm, k-SCST. GUTS is the first German unsupervised ATS approach to the author's knowledge. GUTS can further be applied to any text corpus, domain and language.

While many simplification phenomena happen on a paragraph-level [Alva-Manchego et al., 2019], most of the previous research on ATS has been performed on a sentence-level. Simplifying individual sentences is an easier task since the space of possible simplifications increases with the length of the text. With this work, a paragraph-level simplification system for German is presented that is able to simplify texts of varying sizes.

Further contributions of this thesis are changes for individual parts of GUTS compared to KiS. Especially the modifications for individual scores of the reward. With $S_{hallucination}$, a novel hallucination detection method is presented (Section 3.1.4). This score can detect named entities that have been introduced by a text generation model and are not found in the source text. This method is arguably implemented more dynamically than the implementation in KiS [Laban et al., 2021b, §3.4.2]. Their score directly matches the named entities in the source text and the generated text, which may be more prone to detect a false hallucination. However, their score also identifies false and hallucinated numeric values that $S_{hallucination}$ could not do.

The meaning preservation score in this work (Section 3.1.2) also differs from the method of KiS. They use a model to directly predict keywords based on the simplification, which forms the corresponding score for the reward [Laban et al., 2021b, §3.3]. The score in this work presents a combination of sentence alignment and similarity measuring using

BERTScore [Zhang et al., 2019], to rate how well the content of the original paragraph is preserved in the generated simplification. Both approaches showed their benefits and disadvantages discussed in Section 5.2.2.

Lastly, an article repetition penalty was introduced as an additional score to the reward. Since the German generator model, GerPT-2 cheated the scores by repeating German articles (e.g. "der", "die"). After the addition of this score, this behaviour of the generator was unlearned. These before-mentioned scores can also be used for other text generation tasks.

6.2 Limitations

Different problems and limitations have been found during the analysis of rewards, the conducted experiments, and the evaluation of GUTS. Due to time restrictions, some of these problems could not be sufficiently analysed. For further exploration of these issues, more parameters and settings need to be explored. More and longer experiments need to be conducted to explain the following limitations better.

The reference corpus TextComplexityDE offered only a limited amount of simplification data. Also, this dataset consists of sentence simplifications. Since GUTS learned on paragraph-level data, the dataset needed to be manually adapted. The design and analysis of the reward scores in this work relied entirely on this dataset. This was a limiting factor for the quality of the reward for GUTS.

Since KiS used the large English simplification dataset Newsela for reference, their reward and settings could only be used loosely as a reference. Some of their methods would not work with the German language. One reason for this is that the German language is arguably grammatically more complex and morphologically richer than English.

During training and evaluation of the model's outputs, several limitations of GUTS have been identified.

Simplification phenomena: During the manual evaluation, some lexical and syntactic simplifications of good quality have been observed. Good lexical simplifications, like substitution with simpler synonyms, were rarely observed. Increasing the weight of $S_{lexical}$ for the reward primarily led to word length reductions that were often undesirable. These word reductions would often result in information loss, faithfulness issues and ungrammatical texts. These observations indicate that the score for lexical simplicity might not be sufficient for German lexical simplifications. The score solely relied on relative word frequencies to determine the complexity of words. The method seemed vulnerable to being cheated by the generator model during training.

While the simplicity of the sentence structures according to $S_{syntactic}$ improved with the GUTS models, this metric seemed to be too elementary to learn high-quality syntactic simplifications consistently. The syntactic changes mainly consisted of sentence/phrase deletions, sometimes sentence joining and very rarely sentence splitting operations. More experiments and analyses need to go into this aspect to confirm the limitations and capabilities of this score.

Fluency: Grammatical mistakes and non-fluent samples during the experiments were an issue in this work. The arguably best model from this work, GUTS-2, outputs mostly fluent texts, with primarily minor grammatical errors like confusing

Model	Function	GPU (Index)	Size
GerPT-2	Generator	0	643MB
Sentence-Transformer	$S_{meaning}$	1	514MB
BERT base	$S_{fluency}$	0	422MB
BERT base	$S_{meaning}$, $S_{guardrail}$	1	422MB
BERT base	S_{discr}	1	422MB
RoBERTa (NER)	$S_{hallucination}$	1	1030MB
			3453MB

Table 9: Overview of transformer models used for GUTS

articles. Nonetheless, this is one of the most important criteria and needs to be reliable for a sufficient ATS system. There might be various reasons that caused these issues. One is the suitability of the reward scores assessing the fluency of the generated simplifications. Especially the language model-based fluency score highly rated some non-fluent patterns. An additional reason might be the choice of the decoding method, which sets the quality for the sampled simplification candidates.

Faithfulness: Non-factual content in the produced simplifications was another dominant issue with GUTS. This limitation is an ongoing research field for text generation tasks, such as summarization [Fischer, 2021; Cao et al., 2018; Falke, 2019]. The GUTS models regularly generated simplifications with factual incorrect numbers and dates. Furthermore, the models sometimes introduced hallucinations to the simplifications, which led to disinformation.

Most of these limitations cannot be directly traced back to a single cause. The training signal with this approach relies entirely on the quality of the generated samples and the reward. Since multiple scores and sampling problems can be a factor in the issues, more work needs to be done in order to explore and analyse these issues.

Another issue, briefly touched on in Section 4.2, was the memory and speed limitations experienced during training. The GUTS framework contained multiple large transformer models. The size and number of these models during training presented some challenges.

For these models to run at an acceptable speed, they needed to be executed on a GPU. Table 9 shows the size of all models and the index of the GPU they were executed for during training. Note that these are the raw model sizes. When processing data, the memory consumption increases accordingly. During training, OutOfMemory (OOM) errors were repeatedly experienced, signalling that the memory consumption exceeds the given memory capacities of the GPU. The training runs were performed on two RTX 2080 Ti with 11GB RAM. During development, different models were tested for the individual scores of the reward. Various available German or multilingual models were considered to find the models with a good trade-off between size and performance. The generator and one BERT model were allocated on one GPU, with all other models on the second. Even with this distribution, the input paragraphs during training needed to be limited to 175 tokens to limit the memory consumption further.

Sampling simplification candidates from the generator was another challenge regarding

memory. With KiS, the sampling was performed with caching: After a new word w_i has been sampled, the previously computed attention values of the generator model are saved. Thus only the last generated word w_i is needed as an input to generate the following word w_{i+1}. The information of the input paragraph and the previously generated words $w_{<i}$ are contained in the saved attention values. This decreases the computation time at the cost of memory consumption by the past attention values. [Laban et al., 2021b] The longer the input sequence processed by the generator, the more memory is allocated. Alternatively, the sampling can be performed without saving the attention values. The whole input paragraph and the previously generated words $w_{<i}$ need to be inputted to produce each new word. This method increases the computation time but has a relatively low memory consumption. In this work, both methods have been tested. The available memory of the GPUs was insufficient, which is why the second method has been used. This way, memory was saved during the training process at the cost of speed.

The combination of using multiple large transformer models and the computation-intensive sampling resulted in a relatively slow training speed.

6.3 Future Work

With this work of a German unsupervised simplification system, the basis for further improvements and adaptation has been presented. The GUTS framework consists of multiple components, each of them can be fine-tuned, adapted, or swapped out. The problems and limitations have already been outlined in this chapter. The following contains aspects of this work that can be improved in future work.

The low amount of good lexical simplifications is attributable to the Zipf-based lexical score. The generator cheated the score during the experiments, leading to unwanted behaviour. For improvements of this aspect, a more sophisticated approach might be needed. A score measuring the lexical simplicity should encourage the substitution with simpler synonyms while minimising the room to cheat for the generator. A direction one could follow is considering the German paraphrase database (PPDB) [Ganitkevitch and Callison-Burch, 2014], a collection of 28 000 000 paraphrase pairs, to improve the score or support the generator. Qiang et al. [2021] have shown an approach that uses Zipf to measure word frequency and the PPDB, among other things, for lexical simplification.

Similar problems were observed with syntactic simplicity. While some good structural simplicity phenomena were found with the results, the generator model cheated the FRE in some ways. Tanprasert and Kauchak [2021] already highlighted how metrics, like FKGL and FRE, are too easily exploitable and were created to rate human-written texts. Going forward, $S_{syntactic}$ can further be improved to make it harder to exploit.

In the KiS paper, they already labelled their hallucination detection as rudimentary [Laban et al., 2021b, §3.4.2]. The corresponding score in the GUTS framework $S_{hallucination}$ worked similarly. One big difference is that their score considers numeric values. The GUTS model consistently generated the wrong dates and numeric values since it was not penalized by the score. This aspect can be adopted from the KiS approach for future improvements. Moreover, both the KiS and GUTS models still produced other factual incorrect simplifications. One could adapt recent methods regarding faithfulness from the summarization domain to combat such issues further [Kryściński et al., 2019; Fischer, 2021; Cao et al., 2018].

Measuring the fluency or grammatical correctness of language is an ongoing field of research. There has been little work published on this subject for German at the time of writing this thesis. The LM-fluency score has been shown to create high score returns for simplifications with grammatical mistakes. This could be improved by optimising the current LM-based approach by changing the formula calculating the score from the LM-loss, using a different model, or fine-tuning it on a different dataset. Additionally, automatic grammar correction could be added to the model to improve the generated texts [Ge et al., 2019, see].

Another essential part of the training is the sampling of simplification candidates. Choosing a decoding method and selecting optimal parameters is highly dependent on the given task. Further testing and parameter optimisation is encouraged to increase the candidates' quality and diversity. Having better samples can ultimately lead to faster and more robust training and decrease the overall problems with the system.

Laban et al. [2021b, §6] brought up the idea of using this approach as a simplification pre-training strategy or training jointly with supervised training. This idea is especially interesting for the German language since only a few small parallel simplification datasets exist.

A human-evaluation study was missing entirely in this thesis. Creating a study for qualitative evaluation of the simplification system would be an important addition to this work. Comparing this with different models will give more insight into the performance of GUTS, which can be used for further improvements of the whole framework.

Orienting on German simplification guidelines to design the scores for the reward and use for the evaluation would add a qualitative reference to the approach. Rules like this are compiled in "Barrierefreie-Informationstechnik-Verordnung" [2019] or by Maaß [2015].

Overall the GUTS framework has several settings and hyperparameters that can be explored and optimised: Tweaking the learning rate, updating the scoring function for the reward and modifying the weights of the reward's scores.

Lastly, it is encouraged to try this approach out for different domains and languages. This can be adapted by changing the models and dataset to train on. Even a multilingual simplification approach would be feasible with this method.

Chapter 7

Conclusion

This thesis has presented GUTS, an approach for unsupervised text simplification on a paragraph-level for German texts. For this, the work from Laban et al. [2021b] was taken and adapted. Various parts of their implementation have been reworked. Different methods for the individual reward scores have been analysed and compared. With it, different areas of NLP have been explored, such as measuring lexical and syntactic simplification, rating the fluency of generated texts or detecting hallucinations. Most of these had to be changed to work with the new domain and language or to sidestep insufficient data. Several decoding strategies have been critically examined for sampling simplification candidates from the generator model. These insights can be used to improve the training algorithm k-SCST further. Two GUTS models were trained using k-SCST to simplify German Wikipedia paragraphs. These models and an introduced Pivot model, based on the English simplification approach from Laban et al. [2021b], were evaluated. The automatic results showed insignificant differences between them. Despite the relatively short training time, the manual evaluation revealed first lexical and syntactic simplification phenomena with GUTS. Besides that, some problems have been found regarding faithfulness and fluency. Possible causes for these findings are the insufficiency of the score measuring the fluency of a generated text and the rudimentary hallucination detection method. Further suggestions for the different reward scores and other aspects of the approach are laid out for future work. This thesis sets a foundation for further improvements towards an unsupervised German text simplification method and possible adaptions to different domains and languages.

Bibliography

Adelani, D. (2021). Davlan/xlm-roberta-base-ner-hrl. `https://huggingface.co/Davlan/xlm-roberta-base-ner-hrl`. Last visited on April 24, 2022.

Agrawal, S. and Carpuat, M. (2019). Controlling text complexity in neural machine translation. *CoRR*, abs/1911.00835.

Al-Thanyyan, S. S. and Azmi, A. M. (2021). Automated text simplification: A survey. *ACM Computing Surveys (CSUR)*, 54(2):1–36.

Alva-Manchego, F., Scarton, C., and Specia, L. (2021). The (un) suitability of automatic evaluation metrics for text simplification. *Computational Linguistics*, 47(4):861–889.

Alva-Manchego, F. E., Scarton, C., and Specia, L. (2019). Cross-sentence transformations in text simplification. In *WNLP@ ACL*, pages 181–184.

Amstad, T. (1978). *Wie verständlich sind unsere Zeitungen?* Studenten-Schreib-Service.

Attardi, G. (2015). Wikiextractor. `https://github.com/attardi/wikiextractor`.

Battisti, A. and Ebling, S. (2019). A corpus for automatic readability assessment and text simplification of german. *CoRR*, abs/1909.09067.

Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com.

Breland, H. M. (1996). Word frequency and word difficulty: A comparison of counts in four corpora. *Psychological Science*, 7(2):96–99.

Bundesministerium der Justiz (2019). Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie-Informationstechnik-Verordnung - BITV 2.0). `https://www.gesetze-im-internet.de/bitv_2_0/BITV_2.0.pdf`. Last visited on April 24, 2022.

Campbell, M., Hoane, A., and hsiung Hsu, F. (2002). Deep blue. *Artificial Intelligence*, 134(1):57–83.

Cao, M., Dong, Y., Wu, J., and Cheung, J. C. K. (2020). Factual error correction for abstractive summarization models. *arXiv preprint arXiv:2010.08712*.

Cao, Z., Wei, F., Li, W., and Li, S. (2018). Faithful to the original: Fact aware neural abstractive summarization. In *thirty-second AAAI conference on artificial intelligence*.

Carroll, J., Minnen, G., Canning, Y., Devlin, S., and Tait, J. (1998). Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10. Citeseer.

Carroll, J. A., Minnen, G., Pearce, D., Canning, Y., Devlin, S., and Tait, J. (1999). Simplifying text for language-impaired readers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 269–270.

Celikyilmaz, A., Bosselut, A., He, X., and Choi, Y. (2018). Deep communicating agents for abstractive summarization. *arXiv preprint arXiv:1803.10357*.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Chan, B., Möller, T., Pietsch, M., and Soni, T. (2021). bert-base-german-cased. `https://huggingface.co/bert-base-german-cased`. Last visited on April 24, 2022.

Chandrasekar, R., Doran, C., and Bangalore, S. (1996). Motivations and methods for text simplification. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

Choshen, L., Fox, L., Aizenbud, Z., and Abend, O. (2019). On the weaknesses of reinforcement learning for neural machine translation. *CoRR*, abs/1907.01752.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, page 138–145.

Evans, R., Orasan, C., and Dornescu, I. (2014). An evaluation of syntactic simplification rules for people with autism. Association for Computational Linguistics.

Falke, T. (2019). Automatic structured text summarization with concept maps.

Fan, A., Lewis, M., and Dauphin, Y. N. (2018). Hierarchical neural story generation. *CoRR*, abs/1805.04833.

Fischer, T. (2021). *Finding Factual Inconsistencies in Abstractive Summaries*. PhD thesis, Universität Hamburg.

Flesch, R. (1943). Marks of readable style; a study in adult education. *Teachers College Contributions to Education*.

Flesch, R. (1948). A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Foundation, W. (2022). Wikimedia downloads. `https://dumps.wikimedia.org`. Last visited on April 24, 2022.

Ganitkevitch, J. and Callison-Burch, C. (2014). The multilingual paraphrase database. In *LREC*, pages 4276–4283. Citeseer.

Garg, S., Peitz, S., Nallasamy, U., and Paulik, M. (2019). Jointly learning to align and translate with transformer models. *arXiv preprint arXiv:1909.02074*.

Ge, T., Zhang, X., Wei, F., and Zhou, M. (2019). Automatic grammatical error correction for sequence-to-sequence text generation: An empirical study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6059–6064.

Gemeinnützige Werkstätten und Wohnstätten - GWW (2022). Gemeinnützige Werkstätten und Wohnstätten - GWW. `https://www.gww-netz.de/de/`. Last visited on April 24, 2022.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Gunning, R. et al. (1952). Technique of clear writing.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network (2015). *arXiv preprint arXiv:1503.02531*, 2.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Holtzman, A., Buys, J., Forbes, M., and Choi, Y. (2019). The curious case of neural text degeneration. *CoRR*, abs/1904.09751.

Kann, K., Rothe, S., and Filippova, K. (2018). Sentence-level fluency evaluation: References help, but can be spared! *arXiv preprint arXiv:1809.08731*.

Keskisärkkä, R. (2012). Automatic text simplification via synonym replacement.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klaper, D., Ebling, S., and Volk, M. (2013). Building a german/simple german parallel corpus for automatic text simplification.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Koto, F., Lau, J. H., and Baldwin, T. (2020). FFCI: A framework for interpretable automatic evaluation of summarization. *CoRR*, abs/2011.13662.

Kryściński, W., McCann, B., Xiong, C., and Socher, R. (2019). Evaluating the factual consistency of abstractive text summarization. *arXiv preprint arXiv:1910.12840*.

Laban, P., Hsi, A., Canny, J., and Hearst, M. A. (2021a). The summary loop: Learning to write abstractive summaries without examples. *arXiv preprint arXiv:2105.05361*.

Laban, P., Schnabel, T., Bennett, P., and Hearst, M. A. (2021b). Keep it simple: Unsupervised simplification of multi-paragraph text. *arXiv preprint arXiv:2107.03444*.

Lau, J. H., Clark, A., and Lappin, S. (2017). Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive Science*, 41(5):1202–1241.

Li, W. (2002). Zipf's law everywhere. *Glottometrics*, 5(2002):14–21.

Li, Y. (2017). Deep reinforcement learning: An overview. *CoRR*, abs/1701.07274.

Lin, C.-Y. and Hovy, E. (2003). Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 human language technology conference of the North American chapter of the association for computational linguistics*, pages 150–157.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Loper, E. and Bird, S. (2002). NLTK: the natural language toolkit. *CoRR*, cs.CL/0205028.

Loshchilov, I. and Hutter, F. (2017). Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Maaß, C. (2015). *Leichte Sprache. Das Regelbuch*. Lit-Verlag.

Mallinson, J., Sennrich, R., and Lapata, M. (2020). Zero-shot crosslingual sentence simplification. Association for Computational Linguistics.

Martin, L., Sagot, B., de la Clergerie, É., and Bordes, A. (2019). Controllable sentence simplification. *CoRR*, abs/1910.02677.

Maynez, J., Narayan, S., Bohnet, B., and McDonald, R. (2020). On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*.

Meister, C., Pimentel, T., Wiher, G., and Cotterell, R. (2022). Typical decoding for natural language generation. *arXiv preprint arXiv:2202.00666*.

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. (2017). Mixed precision training. *arXiv preprint arXiv:1710.03740*.

Minixhofer, B. (2020). GerPT2: German large and small versions of GPT2.

Naderi, B., Mohtaj, S., Ensikat, K., and Möller, S. (2019). Subjective assessment of text complexity: A dataset for german language.

Nakamachi, A., Kajiwara, T., and Arase, Y. (2020). Text simplification with reinforcement learning using supervised rewards on grammaticality, meaning preservation, and simplicity. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 153–159, Suzhou, China. Association for Computational Linguistics.

Narayan, S., Cohen, S. B., and Lapata, M. (2018). Ranking sentences for extractive summarization with reinforcement learning. *CoRR*, abs/1802.08636.

Ng, A. Y. (2003). *Shaping and policy search in reinforcement learning*. PhD thesis.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Petersen, S. E. and Ostendorf, M. (2007). Text simplification for language learners: a corpus analysis. In *Workshop on speech and language technology in education*. Citeseer.

Qiang, J., Li, Y., Zhu, Y., Yuan, Y., Shi, Y., and Wu, X. (2021). Lsbert: Lexical simplification based on bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3064–3076.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rajaraman, A. and Ullman, J. D. (2011). Data mining (pp. 1–17).

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Reimers, N. and Gurevych, I. (2021). sentence-transformers/msmarco-distilbert-multilingual-en-de-v2-tmp-lng-aligned. `https://huggingface.co/sentence-transformers/msmarco-distilbert-multilingual-en-de-v2-tmp-lng-aligned`. Last visited on April 24, 2022.

Rello, L., Baeza-Yates, R., Bott, S., and Saggion, H. (2013). Simplify or help? text simplification strategies for people with dyslexia. W4A '13, New York, NY, USA. Association for Computing Machinery.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rios, A., Spring, N., Kew, T., Kostrzewa, M., Säuberli, A., Müller, M., and Ebling, S. (2021). A new dataset and efficient baselines for document-level text simplification in german. pages 152–161.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Saggion, H. (2017). Automatic text simplification. *Synthesis Lectures on Human Language Technologies*, 10(1):1–137.

Salazar, J., Liang, D., Nguyen, T. Q., and Kirchhoff, K. (2019). Pseudolikelihood reranking with masked language models. *CoRR*, abs/1910.14659.

Säuberli, A., Ebling, S., Volk, M., Gala, N., and Wilkens, R. (2020). Benchmarking data-driven automatic text simplification for german.

Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B., and Staiano, J. (2020). Mlsum: The multilingual summarization corpus. *arXiv preprint arXiv:2004.14900*.

Shardlow, M. (2014). A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550:354–359.

Specia, L. (2010). Translating from complex to simplified sentences. In *PROPOR*.

Speer, R., Chin, J., Lin, A., Jewett, S., and Nathan, L. (2018). Luminosoinsight/wordfreq: v2.2.

Sulem, E., Abend, O., and Rappoport, A. (2018a). BLEU is not suitable for the evaluation of text simplification. *CoRR*, abs/1810.05995.

Sulem, E., Abend, O., and Rappoport, A. (2018b). Semantic structural evaluation for text simplification. *CoRR*, abs/1810.05022.

Suter, J., Ebling, S., and Volk, M. (2016). Rule-based automatic text simplification for german.

Tanprasert, T. and Kauchak, D. (2021). Flesch-kincaid is not a text simplification evaluation metric. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, pages 1–14, Online. Association for Computational Linguistics.

Tiedemann, J. and Thottingal, S. (2020). OPUS-MT — Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vijayakumar, A. K., Cogswell, M., Selvaraju, R. R., Sun, Q., Lee, S., Crandall, D., and Batra, D. (2016). Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

von Platen, P. (2020). How to generate text: using different decoding methods for language generation with transformers.

Wang, A., Cho, K., and Lewis, M. (2020). Asking and answering questions to evaluate the factual consistency of summaries. *CoRR*, abs/2004.04228.

Wang, T., Chen, P., Amaral, K. M., and Qiang, J. (2016). An experimental study of LSTM encoder-decoder model for text simplification. *CoRR*, abs/1609.03663.

Watanabe, W. M., Junior, A. C., Uzêda, V. R., Fortes, R. P. d. M., Pardo, T. A. S., and Aluísio, S. M. (2009). Facilita: Reading assistance for low-literacy readers. SIGDOC '09, page 29–36, New York, NY, USA. Association for Computing Machinery.

Wenzek, G., Lachaux, M.-A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A., and Grave, E. (2020). CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

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

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Wu, L., Tian, F., Qin, T., Lai, J., and Liu, T. (2018). A study of reinforcement learning for neural machine translation. *CoRR*, abs/1808.08866.

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

Xu, W., Napoles, C., Pavlick, E., Chen, Q., and Callison-Burch, C. (2016). Optimizing Statistical Machine Translation for Text Simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Zhang, S. and Bansal, M. (2019). Addressing semantic drift in question generation for semi-supervised question answering. *arXiv preprint arXiv:1909.06356*.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Zhang, X. and Lapata, M. (2017). Sentence simplification with deep reinforcement learning. *CoRR*, abs/1703.10931.

Appendix A

Simplification Examples

A.1 Pivot Model

This figure shows an example for a simplification using the Pivot model, with the translation from German to English, the simplification in English and the translation back to German.

Original (German): Bis etwa Mitte der 1930er-Jahre war die Gitarre ein ausschließlich akustisches Instrument. Die akustische Gitarre ist, verglichen mit vielen anderen akustisch gespielten Instrumenten (zum Beispiel Trompete, Klavier, Schlagzeug), ein eher leises Instrument. Darum ging der Klang der akustischen Gitarre in größeren Ensembles (wie den zu der Zeit sehr populären Big Bands) sehr leicht unter. Aus diesem Grund wurden Gitarristen in Big Bands hauptsächlich zur Begleitung (in einer Rhythmusgruppe) eingesetzt. Als Pionier dieser Funktion gilt Freddie Green, der in der "„All American Rhythm Section“" der Count-Basie-Big-Band spielte.

Original (English): Until the mid-1930s, the guitar was an only acoustic instrument. Compared to many other acoustic instruments (for example trumpet, piano, percussion), the acoustic guitar is a rather quiet instrument. That's why the sound of the acoustic guitar in larger ensembles (like the at the time very popular Big Bands) was very easily lost. For this reason, guitarists in big bands were mainly used for accompaniment (in a rhythm group). Freddie Green, who played in the "All American Rhythm Section" of the Count Basie Big Band, is considered a pioneer of this function.

Simplification (English): Until the mid-1930s, the guitar was an acoustic instrument. Compared to many other acoustic instruments, the acoustic guitar is quiet and simple. That's why it's used in larger ensembles (like the Big Bands) and why it's so easy to play (in rhythm groups) than other instruments. That's why Freddie Green, who played in the Count Basie rhythm section, was considered a pioneer of this function.

Simplification (German): Bis Mitte der 1930er Jahre war die Gitarre ein akustisches Instrument. Im Vergleich zu vielen anderen akustischen Instrumenten ist die Akustikgitarre leise und einfach. Deshalb wird sie in größeren Ensembles (wie den Big Bands) verwendet und deshalb ist es so einfach zu spielen (in Rhythmusgruppen) als andere Instrumente. Deshalb galt Freddie Green, die in der Rhythmusgruppe Count Basie spielte, als Pionier dieser Funktion.

Figure 32: Pivot model example

A.2 Model Simplifications

The following figures show simplifications examples from all three evaluated models: GUTS-1, GUTS-2, and the Pivot model based on [Laban et al., 2021b]. All simplifications have been retrieved from the models using greedy decoding.

Original: Halvarsson gewann bei der Junioreneuropameisterschaft 1979 in Achenkirch die Bronzemedaille im Slalom. Die ersten Punkte in einem Weltcuprennen holte er am 10. März 1980 als Zehnter des Slaloms von Cortina d'Ampezzo. Im selben Jahr gewann er den schwedischen Riesenslalom-Meistertitel. In den zwei folgenden Wintern fuhr er im Weltcup je zwei Mal unter die besten 15, während er beim Slalomrennen der Weltmeisterschaft 1982 ausschied. In der Saison 1982/83 gelangen ihm sechs Platzierungen unter den besten 15, darunter ein sechster Platz in Kranjska Gora.

GUTS-1: Alle Weltmeisterschaften 1979 in Achenkirchen die Bronzemedaillen im Slalom. Der ersten Punkte in einem Wettkampf holte er am 10 März 1980 als Zehntel des Slalom waren. Im selben Jahr gewannen sie den schwedischen Riesenslaloms den Titel. In den zwei darauffolgenden Wintern fuhr er den drei Runden und waren dabei beim Slalom. In sich waren sechs Platzierungen unter den 15 und Platz waren Platz.

GUTS-2: Hvar gewann bei der Junioren Europameisterschaft 1979 in Achenkirchen die Bronzemed Medaille im Slalom. Der ersten Punkte in einem Wettkampf holte er am 10." Bis zum Zehnten des Slaloms" von Cortina d'Ampezzo. Beim selben Jahr gewann er die schwedischen Riesenslalom. In den zwei Jahren fuhr er im Weltcup jeweils zwei Mal unter die guten 15, während er bei beim Slalom gegeneinander ausschied. In den Saison 1982/83 gelang ihm sechs Platzierungen unter dem besten 15, darunter einen sechster Platz in Ku.

Pivot: Halvarsson gewann die Bronzemedaille bei der Junioren-Europameisterschaft 1979. Er war einer der ersten Spieler, der die Slalombarriere durchbrach, und er gewann die ersten beiden Punkte in einem Rennen am 10. März 1980. Zwei Jahre zuvor hatte er den schwedischen Riesen-Slalom gewonnen, aber dieses Mal war er am Rande des Sieges. In den folgenden zwei Wintern gewann er die schwedische Meisterschaft, einschließlich des Slaloms, und zog sich in der Saison 1982/83.

Figure 33: Simplification examples 1

Original: König Dom Manuel I. nahm ab dem Ende des 15. Jahrhunderts weitreichende Neugestaltungen in der Stadt Sintra vor. Ab dem 16. Jahrhundert nahm die Beliebtheit Sintras bei den Oberschichten zu, und adlige Familien errichteten im Kreis Sintra einige Herrenhäuser. Das Erdbeben von Lissabon 1755 rief auch in Sintra erhebliche Zerstörungen hervor, denen in der Folge umfangreiche Arbeiten des Wiederaufbaus folgten. Ende 18. Jahrhundert entstand mit der Textildruckerei und -färberei "Fábrica de Estamparia de Rio de Mouro" die erste industrielle Einrichtung im Kreis.

GUTS-1: König Dom Manuel I nahm ab dem Ende der 15. Jahrhunderts weitreichende Veränderungen in der Stadt Sintras vor. Ab dem 17. Jahrhundert nahm die Popularität Sintras bei den Herrenhäuser zu, und waren die neuen Herrenhäuser. Das Beben von Lissabon 1755 riefen auch in Sintra beträchtlichen Schäden hervor, denen in den Folge umfangreiche Arbeiten des Aufbaus folgten. Ende waren sich über 200 Menschen daran zu einem Neuanfang und wurden von den neuen Maschinen wurden neu.

GUTS-2: König Dom Manuel I nahm ab dem Ende der 15. Jahrhundert weitreichende Neugestaltungen. In der Stadt Sintra ab dem 16. Jahrhundert gab die Beliebtheit Sintras. Ab dem 16 Jahr nahm die Beliebtheit Sintra bei den Oberschichten bei, und adlige Familie errichteten im Kreis Sint. Das Erdbeben von Liss 1755 rief auch noch in Sintra erhebliche Schäden hervor, denen in den Folge umfangreiche Arbeiten des Aufbaus folgten. Ende 19. Jahrhundert entstand mit dem Textilddruckerei und -firberei "Fébrica de Estaparia de Rio de" die erste industrielle Organisation im Kreis.

Pivot: Seit dem Ende des 15. Jahrhunderts wurde Sintra umfassend renoviert. Ab dem 16. Jahrhundert war die Stadt für ihre hohe Lebensqualität bekannt geworden. Von dort aus begannen Adelsfamilien, im Kreis Sintra Villen zu bauen, die eine große Anzahl von Geschäften und Restaurants umfassten. Das Erdbeben in Lissabon im Jahr 1755 verursachte auch erhebliche Schäden, was zu umfangreichen Wiederaufbauarbeiten führte. Ende des 18. Jahrhunderts wurde die erste Industrieanlage in der Gegend

Figure 34: Simplification examples 2

Original: Mizuo Peck wuchs in New York auf und stand bereits in jungen Jahren auf der Theaterbühne. Sie besuchte die Theaterabteilung der State University of New York und machte ihr Filmdebüt im Jahr 2000 in dem deutschen Fernsehfilm "Nicht heulen, Husky" neben Heiner Lauterbach und Barbara Rudnik. Bekannt wurde sie aber vor allem durch ihre Rolle als Sacajawea in der dreiteiligen Filmreihe "Nachts im Museum" zwischen 2006 und 2014.

GUTS-1: Während zuo Peck wuchs und stand bereits in junge Jahre auf der Theaterbühne und machte ihr Film debüt im Jahr 2000. Sie besuchte die Schule und machte ihr Filmvideo im Jahr 2000 und wurde von den Schauspielern sind über den Fehler gesprochen. Bekannt wurde sie war aber vor allem durch ihr Rolle als Sacajawa in der dreiteilsigen Filmreihe. Zwischen 2006 und 2014.

GUTS-2: Mizuo Pecks wuchs in New York und stand bereits in den jungen Jahren auf der Bühne. Sie besuchte die Schauspielabteilung der State University von New York und machte sich ihr Filmdebüt. Bekannt wurde sie doch vor allem durch ihre Rollen als Sacajawe in der dreiteiligsten Filmreihe "Nacht" zwischen 2006 und 2015.

Pivot: Mizuo Peck wurde in New York geboren und hatte bereits in jungen Jahren eine Karriere als Theaterdarstellerin. Sie besuchte die Theaterabteilung der State University of New York und debütierte Ende 2000 in einer deutschen TV-Serie mit dem Titel „Nicht heulen, Husky". Doch bald wurde sie in ihrem eigenen Film Sacajawea zur Vollzeitdarstellerin,

Figure 35: Simplification examples 3