

---

# Generation of German Text Controlled by Sentiment and Keywords

---

Paulina Aleksandra Żal

Master's Thesis

Master of Science in Applied Information and Data Science

School of Business

Lucerne University of Applied Sciences and Arts

Wettingen AG, 22nd of December 2023

Author:	Paulina Aleksandra Żal paulina.zal@stud.hslu.ch
Supervisor:	Dr. Guang Lu guang.lu@hslu.ch
Co-Supervisor:	Dr. Nianlong Gu nianlong@ini.ethz.ch

# Management Summary

The introduction of transformer models into Natural Language Processing (NLP) led to Pre-trained Language Models (PLM), which are constantly becoming more attractive for a variety of today’s use cases. Due to their power, these models can be employed in Controllable Text Generation (CTG) to exert precise control over attributes like sentiment or topic, advancing text customization in various applications. While proficient, the probabilistic nature of those models may lead to outputs not precisely aligned with user intent. Emerging methods such as Fine-Tuning and Plug-and-Play Models (PPLMs) empower users to steer text generation, aiming for more tailored outputs, particularly in marketing and content creation domains.

This thesis contributes by refining text generation techniques, ensuring outputs adhere to predefined guidelines related to both topic and sentiment. The aim is to modify a pre-trained GPT-2 model in German, enabling it to produce paragraphs derived from an input sequence, a sentiment token, and a specific keyword set.

The pipeline consists of Supervised Fine-Tuning (SFT), optimization with Reinforcement Learning (RL), and a Logit Modification mechanism. SFT is utilized to adapt the GPT-2 model to generate text concerning the specified sentiment token. RL optimizes the model and improves the text generation in terms of sentiment. The model is rewarded by scores from a Sentiment Discriminator that was previously created. In the last step a logit modification mechanism from Pascual et al. (2020) is adapted and tested with different decoding strategies.

SFT and RL show promising results in generating sentiment-based text. However, implementing keyword control mechanisms reduces the performance in terms of keyword utilization compared to the GPT-2. The human assessment indicates that texts generated with sentiment tokens exhibit moderate fluency and coherence. The introduction of keyword control negatively impacts fluency and coherence. Using the method may save time and effort during the creative process of content creation. However, the proposed pipeline is not yet suitable for practice, as it faces some limitations in generating coherent and fluent texts with a specific set of keywords. The author believes that keyword control could be improved within the SFT and RL processes to achieve a balanced emphasis on both sentiment and keywords in Fine-Tuning models.

**Keywords.** Natural Language Processing, Controllable Text Generation, Pre-trained Language Models, Supervised Fine-Tuning, Reinforcement Learning

# Contents

<b>List of Figures</b>	<b>6</b>
<b>List of Tables</b>	<b>7</b>
<b>List of Code</b>	<b>9</b>
<b>List of Abbreviations</b>	<b>10</b>
<b>Acknowledgements</b>	<b>11</b>
<b>1 Introduction</b>	<b>12</b>
1.1 Background . . . . .	12
1.2 Topic Definition and Thesis Objective . . . . .	13
1.3 Related Work . . . . .	14
1.4 Research Questions . . . . .	15
1.5 Structure of this Thesis . . . . .	16
<b>2 Theoretical Fundamentals</b>	<b>16</b>
2.1 Transformers Architecture . . . . .	16
2.1.1 Attention Mechanism . . . . .	18
2.1.2 Encoder . . . . .	19
2.1.3 Decoder . . . . .	19
2.2 Pre-trained Language Models . . . . .	20
2.2.1 GPT . . . . .	20
2.2.2 BERT . . . . .	20
2.3 Text Generation . . . . .	21
2.3.1 Controllable Text Generation . . . . .	22
2.3.2 Plug-and-Play Language Models . . . . .	23
2.4 Decoding Strategies . . . . .	24
2.4.1 Greedy Search . . . . .	24
2.4.2 Top- $k$ . . . . .	24
2.4.3 Top- $p$ . . . . .	24
2.4.4 Beam Search . . . . .	25
2.5 Reinforcement Learning . . . . .	26
<b>3 Research Design and Data</b>	<b>27</b>
3.1 Workflow . . . . .	27
3.2 Data . . . . .	28
3.2.1 Dataset Description . . . . .	28

3.2.2	Data Cleaning . . . . .	29
3.3	Automated Evaluation Metrics . . . . .	30
3.3.1	Classification Metrics . . . . .	30
3.3.2	Perplexity . . . . .	31
3.3.3	SLOR . . . . .	32
3.3.4	Flesch Reading Ease . . . . .	32
3.3.5	Coherence Score . . . . .	32
3.3.6	Success Rate . . . . .	33
3.4	Sentiment Classification . . . . .	33
3.5	Fine-Tuning . . . . .	34
3.6	Proximal Policy Optimization . . . . .	34
3.7	Logits Modification Mechanism . . . . .	36
3.8	Human Evaluation . . . . .	37
<b>4</b>	<b>Experiments</b>	<b>38</b>
4.1	Data Combination and Preprocessing . . . . .	39
4.2	Sentiment Discriminator . . . . .	39
4.2.1	Data Processing . . . . .	39
4.2.2	BERT model with Convolutional Neural Network . . . . .	40
4.2.3	BERT model with Classification Head . . . . .	41
4.3	Sentiment Control . . . . .	42
4.3.1	Supervised Fine-Tuning . . . . .	42
4.3.2	Reinforcement Learning Optimization . . . . .	43
4.4	Keyword Control . . . . .	46
4.4.1	Logit Modification Mechanism . . . . .	46
4.4.2	Decoding Strategies . . . . .	47
4.5	Survey Design . . . . .	48
<b>5</b>	<b>Results</b>	<b>50</b>
5.1	Performance of Sentiment Discriminator . . . . .	51
5.2	Evaluation of Sentiment Control . . . . .	51
5.2.1	Evaluation of Supervised Fine-Tuning . . . . .	52
5.2.2	Results of Reinforcement Learning . . . . .	54
5.2.3	Improvement of Sentiment Control During Training . . . . .	56
5.3	Performance of Keyword Control . . . . .	56
5.3.1	Evaluation Based on Nouns . . . . .	56
5.3.2	Evaluation Based on Sentiment-Carrying Adjectives . . . . .	58
5.3.3	Evaluation of Different Control Inputs . . . . .	59

5.4	Analysis of Survey Results . . . . .	60
5.4.1	Selection of Models . . . . .	60
5.4.2	Demographics . . . . .	60
5.4.3	Inter-Annotator Agreement . . . . .	61
5.4.4	Evaluation of Cronbach Alpha . . . . .	61
5.4.5	Evaluation of Mutual Influence of Keyword and Sentiment Control	61
5.4.6	Influence of Fine-Tuning on Sentiment Control . . . . .	63
5.4.7	Influence of Decoding Strategy and Keyword Control . . . . .	64
<b>6</b>	<b>Discussion</b>	<b>65</b>
6.1	Research Question 1 . . . . .	66
6.2	Research Question 2 . . . . .	67
6.3	Research Question 3 . . . . .	68
6.4	Research Question 4 . . . . .	69
<b>7</b>	<b>Conclusion and Outlook</b>	<b>70</b>
7.1	Conclusion . . . . .	70
7.2	Outlook . . . . .	71
	<b>References</b>	<b>73</b>
	<b>Appendix A Function for Logit Modification</b>	<b>80</b>
	<b>Appendix B Beam Search</b>	<b>82</b>
	<b>Appendix C Examples of Generated Texts</b>	<b>84</b>
	C.1 Texts Generated by Different Models with Negative Sentiment Token .	84
	C.2 Texts Generated by Different Models with Positive Sentiment Token .	85
	<b>Appendix D Survey</b>	<b>87</b>
	<b>Appendix E Texts Evaluated in the Survey</b>	<b>90</b>
	<b>Appendix F Likert Scale Interpretation, Categories and Items</b>	<b>94</b>
	<b>Appendix G Functions for SLOR and Coherence Score</b>	<b>96</b>
	<b>Appendix H Evaluation of Survey's Items</b>	<b>98</b>
	<b>Appendix I Influence of Human Intervention on Text Quality</b>	<b>102</b>
	<b>Declaration of Originality</b>	<b>103</b>
	<b>Declaration of the use of Generative AI</b>	<b>104</b>

# List of Figures

1	Architecture of a Transformers-based Language Model . . . . .	17
2	Transformers' Attention Mechanisms . . . . .	18
3	Structure of a GPT Model . . . . .	21
4	Structure of a BERT Model . . . . .	22
5	Example of Text Generation . . . . .	23
6	Schema of Beam Search . . . . .	25
7	Schema of Reinforcement Learning . . . . .	26
8	Workflow of this Thesis . . . . .	27
9	Confusion Matrix . . . . .	30
10	Finetuning of LM Using Reinforcement Learning . . . . .	35
11	Mechanism of Logits Modification . . . . .	37
12	Sentiment Discriminator Built on BERT with CNN Layers . . . . .	40
13	Sentiment Discriminator Built on BERT with Classification Head. . . .	41
14	Confusion Matrices for Discriminator 1-4 . . . . .	52
15	Trends of Reward Function . . . . .	55
16	Effect of Training (SFT & RL) on Sentiment Control. . . . .	55
17	Demographics . . . . .	61
18	Perception of Text Generated with Sentiment and Keyword Control and GPT-2. . . . .	68
19	Summarized Comparision of Decoding Strategies. . . . .	69
20	Opening Dialog for Demographic Data and Selecting the Path. . . . .	88
21	Example of Text Evaluation. . . . .	89
22	Can Additional Human Intervention Improve the Text Quality? . . . .	102

# List of Tables

1	Results of the Analysis of Datasets. . . . .	28
2	Survey Design. . . . .	49
3	Performance of Sentiment Discriminators. . . . .	51
4	Quality of Texts Created with SFTn Models. . . . .	53
5	Sentiment of Texts Created with SFTn Models. . . . .	53
6	Quality of Texts Created with SFTnRLm Models. . . . .	54
7	Comparision of Sentiment Scores among GPT-2, SFT1 and SFT1RL1. . . . .	54
8	Perplexity of Texts Decoded with Different Strategies and Forcing of Nouns. . . . .	56
9	Influence of Decoding Strategy on Success Rate . . . . .	57
10	Sentiment Accuracy for Different Decoding Strategies. . . . .	58
11	Mutual Performance of Keyword Control with Sentiment-Carrying Words and Sentiment Control. . . . .	59
12	Comparision of Different Control Inputs. . . . .	60
13	Sentiment Perception of Texts Generated with Sentiment Token and Sentiment - Carrying Keywords. . . . .	61
14	Influence of Sentiment Token and Sentiment - Carrying Keywords on Fluency Perception. . . . .	62
15	Human Evaluation of Coherence of Texts Generated with Sentiment Token and Sentiment - Carrying Keywords. . . . .	62
16	Perception of Sentiment of Texts Generated by Models Controlled with Sentiment Token. . . . .	63
17	Fluency's Perception of Texts Generated by Models Controlled with Sentiment Token. . . . .	63
18	Perception of Influence of Fine-Tuning on Coherence. . . . .	64
19	Sentiments Perception of Texts Decoded with Different Strategies. . . . .	65
20	Influence of Decoding Strategies on the Perception of Generated Texts. . . . .	65
21	Influence on Coherence's Perception of Texts Decoded with Different Strategies. . . . .	65
22	Explanation of Likert Scale. . . . .	94
23	Category and Items in the Survey. . . . .	95
24	The Text is Positive. . . . .	98
25	This Text is Free of Spelling and Grammatical Errors. . . . .	99
26	This Text is Well Structured. . . . .	99

27	This Text is Easy to Understand. . . . .	100
28	A Native Speaker Could Have Written the Text Exactly the Same Way.	100
29	This Text Makes Sense. . . . .	101
30	The Transitions in the Text are Well Written. . . . .	101



## List of Code

1	Creation of Instructions for SFT. . . . .	42
2	SFT Training. . . . .	43
3	PPO Model Initialization. . . . .	44
4	Calculation of Model's Reward. . . . .	44
5	Logit Modification Mechanism. . . . .	80
6	Part of Proposed Beam Search. . . . .	82
7	SLOR function. . . . .	96
8	Coherence Score. . . . .	97

## List of Abbreviations

<b>AE</b>	Auto-Encoding
<b>AI</b>	Artificial Intelligence
<b>AR</b>	Auto-Regressive
<b>BERT</b>	Bidirectional Encoder Representations from Transformers
<b>BoW</b>	Bag-of-Words
<b>CNN</b>	Convolutional Neural Network
<b>CTG</b>	Controllable Text Generation
<b>FRE</b>	Flesch Reading Ease Score
<b>GPT</b>	Generative Pre-trained Transformer
<b>IAA</b>	Inter-Annotators Agreement
<b>LDA</b>	Latent Dirichlet Allocation
<b>LLM</b>	Large Language Model
<b>LM</b>	Language Models
<b>ML</b>	Machine Learning
<b>MLM</b>	Masked Language Model
<b>NLG</b>	Natural Language Generation
<b>NLP</b>	Natural Language Processing
<b>NLU</b>	Natural Language Understanding
<b>PLM</b>	Pre-trained Language Model
<b>PPLM</b>	Plug-and-Play Language Model
<b>PPO</b>	Proximal Policy Optimization
<b>ReLU</b>	Rectified Linear Unit
<b>RL</b>	Reinforcement Learning
<b>RNN</b>	Recurrent Neural Network
<b>SFT</b>	Supervised Fine-Tuning
<b>SOTA</b>	State-of-the-art
<b>wPPL</b>	Weighted Perplexity

## Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Guang Lu, for his guidance, time, feedback, and answer to every question. A big thank you goes to Dr. Nianlong Gu for his ideas, suggestions, and discussions.

Finally, my heartfelt appreciation extends to my fiancé, Robby, for your motivation, support, and understanding. Thank you for your patience during those last months.

# 1 Introduction

This Chapter aims to introduce the research behind this master’s thesis. It starts by explaining the background related to text generation in Sect. 1.1. Then in Sect. 1.2, the topic and thesis objective are defined. Section 1.3 presents the work related to this research and is followed by relevant research questions in Sect. 1.4. Section 1.5 closes this Chapter with an overview of the structure of this thesis.

## 1.1 Background

In today’s digital landscape, data takes on a multitude of forms. It can be in the form of text, images, videos, sound, and more. Extracting valuable knowledge and insights from this diverse array of unstructured data sources is a formidable challenge, yet it stands as a fundamental pursuit in fields like data science, Machine Learning (ML), and Artificial Intelligence (AI).

Natural Language Processing (NLP) techniques play a pivotal role in extracting knowledge from textual data. NLP is a collection of computational methods employed to analyze, model, and comprehend human language (Vajjala et al., 2020, p. 4). Notably, recent years have witnessed a remarkable surge in this field, with a transformative moment occurring in 2017. In that year Vaswani et al. (2017) introduced the groundbreaking paper “Attention is all you need”, which unveiled the revolutionary concept of Transformer models. This innovation has reshaped the landscape of NLP and has been instrumental in advancing the understanding and utilization of human language in the digital century.

The advent of this architecture has paved the way for the development of Pre-trained Language Models (PLMs). Prominent among these are the widely recognized models, Generative Pre-trained Transformer (GPT) (Radford et al., 2018) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018). The primary advantage of these models is their pre-training on vast corpora of data, which equips them with a broad understanding of language. State-of-the-art (SOTA) performance can be achieved for downstream tasks with a single requirement of fine-tuning (Zhang et al., 2022). One notable application is Natural Language Generation (NLG). The objective of NLG is to convert diverse data representations into natural language text, aspiring to produce text quality aligning with human-generated content (Reiter & Dale, 2002).

The objective of NLG is to convert diverse data representations into natural language text, aspiring to produce text quality aligning with human-generated content (Reiter

& Dale, 2002). Models that adopt an Auto-Regressive (AR) approach, like the GPT family, excel in this regard. These models predict the next word in a sentence based on the preceding context (Zhang et al., 2022). They are applied in diverse domains, including dialog systems, marketing, and story generation, making NLG a routine part of daily life for people worldwide.

## 1.2 Topic Definition and Thesis Objective

The PLM have opened up vast opportunities for generating contextual text, offering immense potential for a wide range of industries, including e-commerce and marketing. They are capable of learning from the extensive corpora of text data and generating new text that exhibits a generalized understanding of language (Zhang et al., 2022). However, this capability comes with a caveat. The words generated by these models are a product of probabilities, and while they often produce coherent text, they may not consistently align with the precise intent of the user. This probabilistic nature can sometimes lead to unexpected or less controlled outputs, which has led to the pursuit of more controllable and fine-tuned text-generation techniques in the field of NLP.

To address the challenge of controlling text generation, a variety of methods for Controllable Text Generation (CTG) have emerged. These methods primarily fall into three main categories: fine-tuning, retraining, and post-processing. The choice of approach depends on the specific downstream task at hand, as highlighted by Zhang et al. (2022). While these approaches enable CTG, they may still lack open-ended user control over the generated text. In contrast, “plug-and-play” models, as proposed by Zhu et al. (2022), offer a more flexible and interactive way for users to exert control over the text generation process. These models allow users to guide and steer text generation using external inputs, such as keywords or topic labels, providing a more intuitive and user-driven approach to controlling the generated content.

Using these methods to create a text has the potential to contribute to the creation of coherent and engaging textual content or titles. It can be invaluable in marketing efforts. This approach streamlines the creative process of text generation, making it more efficient and expeditious. Moreover, it represents a significant step forward in aligning language models with human language nuances and preferences.

This master’s thesis is dedicated to the domain of CTG in the German language. The objective is to adapt the German GPT-2 model created by Minixhofer et al. (2022). When provided with a sentiment control token, it should generate German text with the specified sentiment. Subsequently, the aim is to employ this adapted model in a way that could generate text based on a set of given keywords. Two key considerations

come into play. First, a sentiment control code is introduced to regulate the emotional tone of the text. Second, a list of keywords is provided to guide the model in generating content with specific terms.

This comprehensive process involves several key steps, including the development of a sentiment discriminator capable of distinguishing between positive and negative sentiments. Additionally, Supervised Fine-Tuning (SFT) on instruction-based text is conducted to prepare the model for sentiment-controlled text generation. The Reinforcement Learning (RL) approach is then employed to further fine-tune the model, making use of the sentiment discriminator to achieve the desired level of sentiment control. Furthermore, a logit modification process (Pascual et al., 2020) is implemented to facilitate the utilization of the specified keywords for content generation.

To evaluate the performance of the adapted model, Perplexity, a common language model metric, is used as a quantitative measure of fluency. Supportive are metrics such as SLOR and Flesch Reading Ease score that allow deeper understanding. In addition, the performance in terms of sentiment and coherence is measured. Lastly, survey participants evaluate whether the generated text meets the expected standards in terms of fluency, sentiment utilization, and coherence. This multifaceted approach aims to create a versatile and controllable text generation approach tailored to specific sentiment and keyword requirements in the German language.

### 1.3 Related Work

Since the introduction of *Transformers* architecture and following the introduction of Language Models (LM) such as GPT (Radford et al., 2018) and BERT (Devlin et al., 2018). The researchers have already established several methods for CTG.

Ziegler et al. (2019) fine-tuned a model with RL from human preferences. Their approach aimed to approximate the dataset’s distribution through reward optimization, incorporating a penalty to maintain the generated text’s fluency closer to the original model. Schulman et al. (2017) proposed a Proximal Policy Optimization (PPO) algorithm that can be also applied in the realm of CTG as shown by von Werra (2023). Post-processing allows the incorporation of controls without extra training (Zhang et al., 2022). A guiding strategy employed within this group is exemplified by the work of Krause et al. (2020). They presented an efficient method that leverages smaller LM to guide the next generation of text, promoting safer and less biased content generation. Their work provides a promising avenue for controlling the output of language models and mitigating issues related to fairness and bias in the generated text. Pascual et al. (2021) presented a simple approach to create text with constraints such as topic or key-

words. Their method, K2T, alters how word probabilities are distributed by favoring words similar in meaning to the specified target word.

To control Text Generation different approaches were presented besides the latter. Pascual et al. (2020) addressed the control of topic generation by stipulating the presence of specific words in the generated text.

He (2021) proposed a model for lexically constrained text generation. It uses a PLM and splits tasks into two sub-tasks. It employs a token-level classifier on the encoder to guide the decoder in replacing and inserting text.

Additionally, researchers put effort into methods that allow topic control. A significant approach was proposed by Dathathri et al. (2019). Their research approach combines a PLM with an attribute classifier, guiding text generation without requiring additional LM training. In addition to topic control, they controlled sentiment.

The sentiment control was also addressed by Hu et al. (2017). They aim to control the sentence sentiment with a token. In their research, they utilized variational auto-encoders and attribute discriminators.

## 1.4 Research Questions

This master’s thesis aims to answer the following questions:

### *Research Question 1*

Can sentiment and keyword control be effectively combined into one model?

### *Research Question 2*

How do users perceive the text, which is generated by the proposed model, compared to a text generated by GPT with respect to a chosen sentiment?

### *Research Question 3*

How is the Perplexity for models with control generation compared to the original GPT?

### *Research Question 4*

What are the differences in Perplexity and keyword utilization using different decoding approaches?

## 1.5 Structure of this Thesis

The following six chapters are divided into Theoretical Background, Methodology, Experiments, Results, Discussion and Conclusions.

Chapter 2 provides a theoretical foundation of concepts on which this master’s thesis is built on. It offers a comprehensive overview of critical concepts relevant to this thesis, beginning with the architecture of transformers and traversing through the text generation process to the principles of RL.

Chapter 3 shows which methods are used in the scope of this research. At first, the dataset and employed evaluation metrics are introduced. Following the key concepts for sentiment and keyword control are presented. The chapter is closed by the presentation of a method for human evaluation.

Chapter 4 shows how the concepts previously presented are implemented in order to create a pipeline that controls text generation.

In Chapter 5, this is followed by the evaluation of the concepts of this pipeline.

Additionally in Sect. 6, the answers to research questions are presented. The last chapter, Chapter 7, discusses the findings of this research and gives an outlook for further work.

## 2 Theoretical Fundamentals

The primary focus of this master’s thesis is devising a potent model for CTG. The forthcoming Chapter encapsulates the pivotal theoretical foundations essential for executing the text generation task.

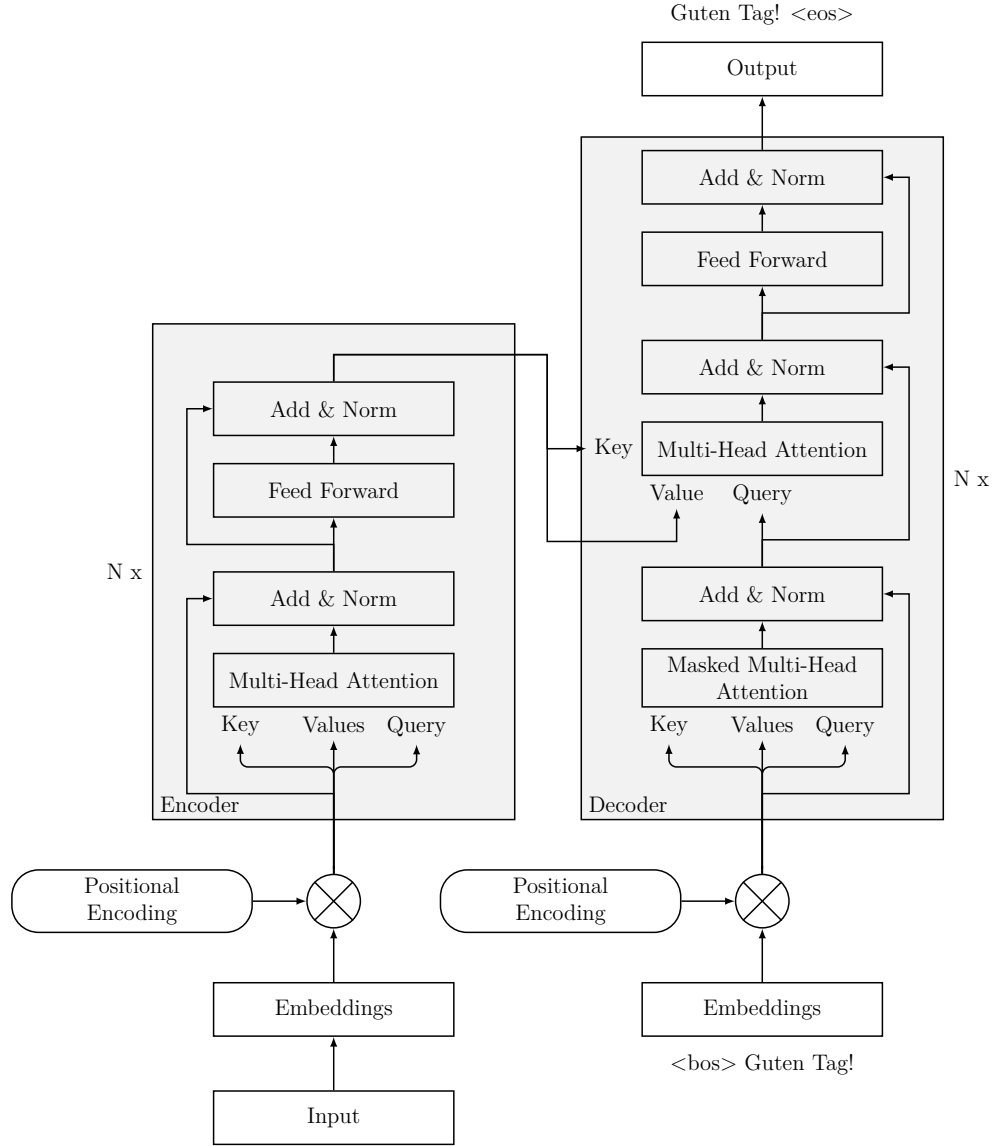
It introduces the Transformers architecture (Sect. 2.1), serving as the foundation for PLM (Sect. 2.2). The process of text generation leverages PLM and is discussed in Sect. 2.3. Additionally, Sect. 2.4 explores decoding strategies for selecting subsequent tokens, while Sect. 2.5 outlines the fundamental principles of RL.

### 2.1 Transformers Architecture

The milestone for building Large Language Model (LLM) was done by Vaswani et al. (2017) with the introduction of the paper “Attention is all you need”. In this paper, the architecture of Transformer models was introduced. The model is shown in Fig. 1. The basis lies in the attention mechanism eliminating the need for recurrence and convolutional layers. The model is built in an encoder-decoder fashion. The encoder



**Figure 1**  
*Architecture of a Transformers-based Language Model*



*Note: Adapted from: Attention is all you need, by Veswani et al., 2017.*

part takes the inputs and transforms them into a continuous representation. The decoder part creates output sequences based on the continuous outputs from the encoder. Unlike Recurrent Neural Network (RNN), which processes text sequentially, the architecture of Transformers operates in parallel (Fan et al., 2020), providing simultaneous predictions for all words. It predicts the next word through iterative sampling from the network's output distribution, a departure from the autoregressive nature of RNNs (Graves, 2013).

**2.1.1 Attention Mechanism.** The primary advantage of the Transformers’ architecture lies within its attention mechanism. Attention-based models can learn meaningful relationships between the words in a sentence by emphasizing the most relevant input tokens at each step (Tunstall et al., 2022, pp. 4–6). The mechanism proposed by Vaswani et al. (2017) can be described mathematically as a function that maps vectors: query and key-value pairs to an output in the form of a matrix. The proposed attention mechanism is created as Scaled Dot-Product Attention. The dot product of each query with all keys is calculated, and then the result is divided by the square root of  $d_k$ , where  $d$  denotes the dimension of keys. After that, a softmax function is used to get the weights that indicate how much attention to pay to each value (Vaswani et al., 2017).

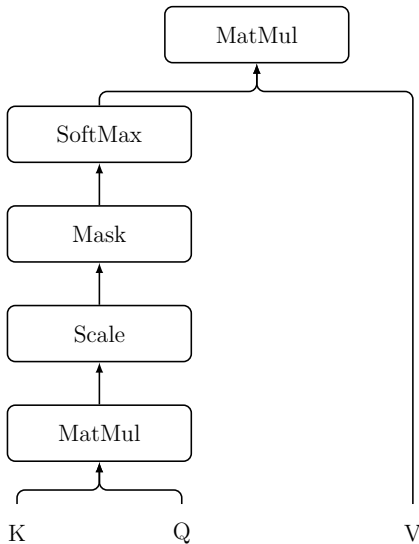
$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k})V \quad (1)$$

In practice, a bunch of queries is merged into a matrix  $Q$ . Bunches of keys and values are put into matrices  $K$  and  $V$  respectively. This allows the calculation of the attention for multiple queries simultaneously as shown in Eq. 1 (Vaswani et al., 2017). The schema of the implementation is depicted in Fig. 2a.

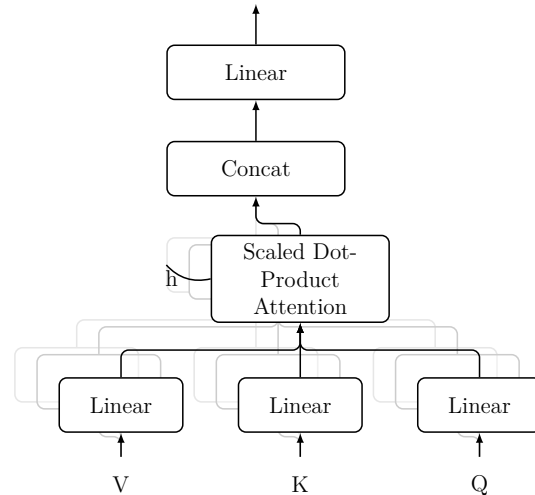
**Figure 2**

*Transformers’ Attention Mechanisms*

(a) *Scaled Dot-Product Attention*



(b) *Multi-Head Attention*



*Note: Adapted from: Attention is all you need, by Vaswani et al., 2017.*

Based on the described attention mechanism by Vaswani et al. (2017), the multi-head attention mechanism (Fig. 2b) was introduced. Vaswani et al. (2017) identified an ad-

vantage in a linear projection of the queries, keys, and values  $h$  times with different, learned linear projections to  $d_q$ ,  $d_k$  and  $d_v$  dimensions, respectively. Multi-head attention allows the model to consider information from various representation subspaces and positions at once Vaswani et al. (2017). The results from each attention head are combined by concatenation, producing a final output vector for every word in the sentence.

**2.1.2 Encoder.** The encoder proposed by Vaswani et al. (2017) consists of  $N = 6$  layers. Every layer consists of multi-head self-attention mechanism and a fully-connected feed-forward network. Each of the sub-layers has a layer normalization on top and is surrounded by residual connection. The first one standardizes every input in the batch to have a zero mean and a standard deviation of one. The skip connections transfer a tensor to the next layer of the model without altering it and then combine it with the processed tensor (Tunstall et al., 2022, pp. 71–73). The network fully connected feed-forward operates independently on each position and applies two linear transformations with a Rectified Linear Unit (ReLU) activation function in between them. The formula for this transformation is denoted as follows:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2)$$

where  $b_i$  is bias neuron in the  $i^{th}$  layer and  $W_i$  is the weight (Vaswani et al., 2017).

The learned embeddings are used to turn input and output tokens into vectors of a specific size. The learned linear transformation and softmax function are applied to convert the decoder’s output into the probabilities for the next predicted token (Vaswani et al., 2017). To make the model aware of the token’s sequence order, Vaswani et al. (2017) introduced positional encodings and combined these to the input embeddings at the bottom of the encoder stack.

**2.1.3 Decoder.** Same as the encoder, the decoder consists of  $N = 6$  layers. Each of these layers consists of three sub-layers. It has two multi-head attention layers and one fully connected feed-forward network. Similarly to the encoder, each of these layers is surrounded by residual connection and has a normalization layer. The first of the multi-head attention layers uses a masking mechanism. This mechanism stops the model from taking into account words that appear after the target word in the sentence (Delovski, 2023). The input of this layer are embeddings of the model’s output combined with the positional encodings (Vaswani et al., 2017). The second multi-head attention layer is initialized with the output from the masked multi-head attention layers and combines

it with the encoder’s hidden states given as keys and values (Tunstall et al., 2022, pp. 76–78).

## 2.2 Pre-trained Language Models

The introduction of the transformer model was the next big step in NLP. Based on this model’s architecture, different PLMs were designed. The PLMs are deep neural networks that are pre-trained on large-scale unlabelled text corpora, which can be further fine-tuned on various downstream tasks (Li et al., 2022), such as Named Entity Recognition (Zhou et al., 2021), Text Classification (Sun et al., 2019) and Text Generation (Li et al., 2022). The best-known examples of PLMs are GPT (Radford et al., 2018) and BERT (Devlin et al., 2018), which utilize the Transformer’s decoder and encoder architecture respectively.

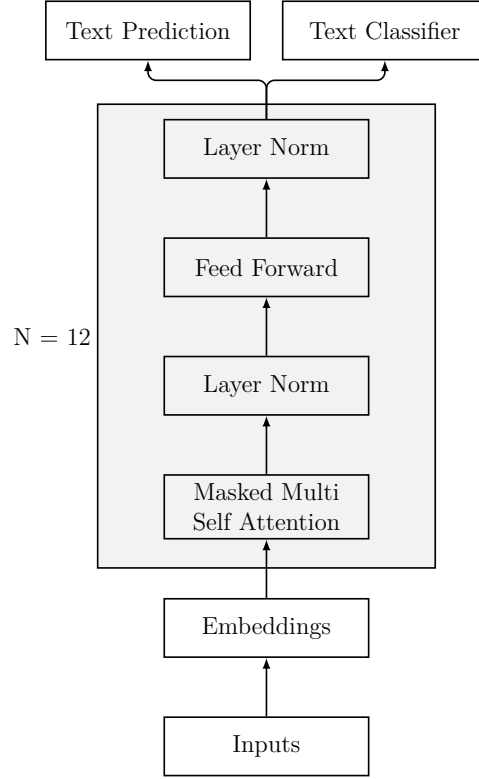
The choice of pre-training objective serves as a significant indicator, of which model to choose. Autoregressive models tend to excel in text generation tasks, while auto-encoders are more adept at comprehending and organizing language, making them suitable for tasks such as sentiment analysis and various information extraction tasks (Lipenkova, 2022).

**2.2.1 GPT.** The GPT model was introduced by researchers from OpenAI, Radford et al. (2018). They used a multi-layer Transformers’ decoder as a language model. The model consists of 12 layers with masked self-attention heads, as shown in Fig. 3. The model was trained for 100 epochs on a large text data corpus containing over 7’000 unpublished books. Most importantly, it has long sections of continuous text, which helps the model to learn from distant information (Radford et al., 2018).

The GPT model is an example of an AR model, which is commonly utilized for text generation. The main task of AR models is to predict the next word based on what has been read in the sequence. During the training phase, a masking mechanism is employed to restrict the attention calculations to consider only the content preceding a word while excluding the content that follows it (Zhang et al., 2022).

**2.2.2 BERT.** The BERT was proposed by Devlin et al. (2018) to pre-train deep bidirectional representations from unlabeled text by simultaneously considering both the left and right context in all layers. As an example of Masked Language Model (MLM), it randomly hides certain tokens from the input and aims to predict the original vocabulary of the concealed word solely based on its context. Alongside the MLM, an additional task is included in Devlin et al. (2018), which predicts whether a pair of

**Figure 3**  
*Structure of a GPT Model*

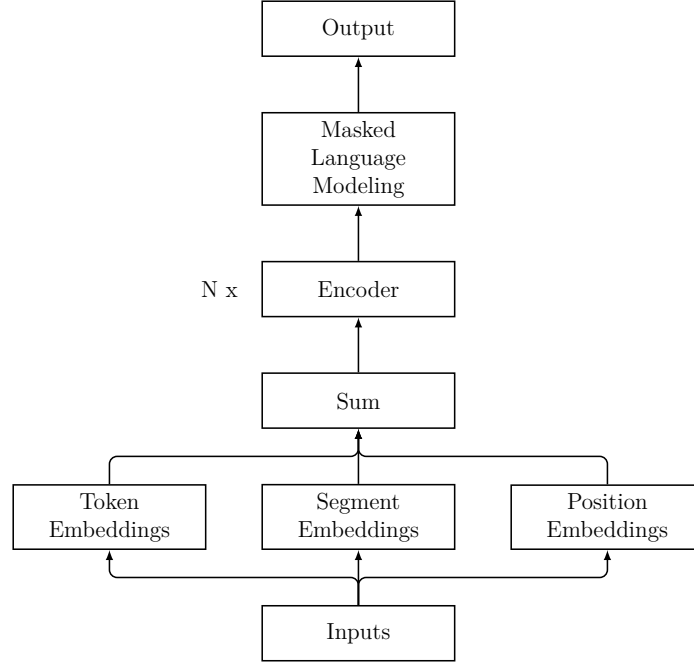


*Note: Adapted from: Improving Language Understanding by Generative Pre-Training, by Redford et al., 2020.*

sentences follow one another. This model consists of the encoder part, with 12 layers (i.e. Transformers blocks). The original, English BERT model was pre-trained on the corpus built on books and Wikipedia (Devlin et al., 2018). As a representative from the Auto-Encoding (AE) Models, it is best suited to tasks related to Natural Language Understanding (NLU), such as text classification and sequence labeling (Zhang et al., 2022). The architecture schema is depicted in Fig. 4.

## 2.3 Text Generation

One of the famous downstream tasks of PLM is text generation. Its primary aim revolves around producing text that closely resembles human language. LLMs learn to predict the next word based on the previous text sequence (Tunstall et al., 2022, pp. 123–126). For text generation, the AR models, like the GPT family are best suited (Zhang et al., 2022). The models learn to predict the probability of  $P(\mathbf{y}|\mathbf{x})$  of  $\mathbf{y} = y_1, y_2, \dots, y_t$  tokens based on the token sequence  $\mathbf{x} = x_1, x_2, \dots, x_k$ , as shown in Eq. 3 (Tunstall et al., 2022, pp. 123–126).

**Figure 4***Structure of a BERT Model*

*Note: Own creation based on: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, by Devlin et al., 2018.*

$$P(y_1, \dots, y_t | \mathbf{x}) = \prod_{t=1}^N P(y_t | y_{<t}, \mathbf{x}) \quad (3)$$

where  $y_{<t}$  denotes  $y_1, y_2, \dots, y_{t-1}$  (Tunstall et al., 2022, pp. 123–127). An example of the text generation process is shown in Fig. 5.

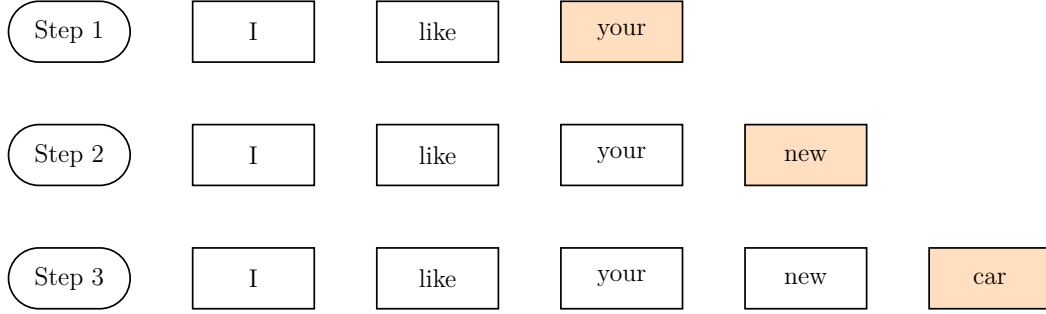
The quality of text generation significantly depends on the appropriate selection of a decoding strategy, which determines the tokens to be used at a specific timestamp. (Tunstall et al., 2022, pp. 123–126). Examples of decoding strategies are Greedy Search, Top- $k$ , Top- $p$ , and Beam Search (see Section 2.4).

**2.3.1 Controllable Text Generation.** Various proposed methodologies aim to steer the LM in different directions for CTG. This area of text generation focuses on generating text, that contains different constraints according to the task-specific wishes of a user. The equation describing the CTG is similar to the general formula for text generation, but it contains  $C$ , which denotes the controlled element (Zhang et al., 2022).

$$P(Y|C) = p(y_1, y_2, \dots, y_t | C) \quad (4)$$

**Figure 5**

Example of the Text Generation process. The newly generated token is depicted in orange.



Note: Adapted from: *Natural Language Processing with Transformers. Building Language Applications with Hugging Face*, by Tunstall et al., 2022, pp. 126.

The CTG can be used for tasks, that require different controlled elements. The controlled aspect can be exemplarily dialog generation and attribute-based generation. As this master’s thesis focuses on the latter, it requires a short explanation.

**2.3.2 Plug-and-Play Language Models.** One step further in improving the user experience with CTG are the Plug-and-Play Language Models (PPLMs). They enable the use of PLMs with control, incorporating specific controlling attributes into the model without necessitating additional fine-tuning or retraining, as highlighted by Dathathri et al. (2019). The main idea is to model the conditional language model  $p(x|a)$ , where  $a$  is an attribute. This can be applied by using the Bayesian formula (Eq. 5).

$$p(x|a) \propto p(a|x)p(x) \quad (5)$$

where  $p(x)$  is a probability distribution over all text and  $p(a|x)$  is the attribute model, which takes a sentence  $x$  and returns the probability of processing the attribute  $a$  (Dathathri et al., 2019).

To control the text generation in a plug-and-play manner, the following approaches have been introduced. Dathathri et al. (2019) controlled the text generation with sentiment discriminator and keywords by using Bag-of-Words (BoW). Krause et al. (2020) tried to control sentiment while preserving the knowledge about the topics. Pascual et al. proposed two approaches for controlling the keywords: Keywords2Text (Pascual et al., 2021) and Directed Beam Search (Pascual et al., 2020).

## 2.4 Decoding Strategies

Decoding methods within text generation outline how a PLM explores potential outputs while generating a new word. As the search space is infinite, the decoding procedures are an important part of the text generation pipeline (Zarrieß et al., 2021). In this section, the most popular decoding strategies will be described.

**2.4.1 Greedy Search.** The straightforward method for choosing the next word is the Greedy Search. This involves selecting the option with the highest probability from the list of potential outputs, determined by the previously calculated probability distribution (López Espejel, 2022). The equation for this approach reads

$$y_t = \operatorname{argmax}_{y \in V} P(y|y_1, y_2 \dots y_{t-1}, x). \quad (6)$$

Because this method greedily selects the best output at each step, its primary drawback becomes evident: the output sequence might begin repeating the same tokens continuously (López Espejel, 2022).

**2.4.2 Top- $k$ .** Another significant strategy is known as Top- $k$ . This approach was detailed by Fan et al. (2018). At each step, the model calculates the probabilities for each word in the vocabulary. Then,  $k$  most likely candidates are selected, resulting in avoiding the low-probability tokens. The probability mass is redistributed among  $k$  candidates. In the subsequent step, the model samples the next word from these candidates (von Platen, 2020).

**2.4.3 Top- $p$ .** An additional crucial decoding strategy was employed by Holtzman et al. (2019). During Top- $p$  (Nucleus Sampling) decoding, words are chosen from a minimal set of words where their aggregated probabilities exceed the threshold probability  $p$ . Subsequently, the probability mass is redistributed among this specific set of words. The original distribution is re-scaled to a new distribution, from which the next word is sampled:

$$P'(x|x_{1:i-1}) = \begin{cases} P(x|x_{1:i-1})/p' & \text{if } x \in V^{(p)} \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

$$\text{and } p' = \sum_{x \in V^{(p)}} P(x|x_{1:i-1}).$$

In practical terms, this involves selecting the most likely tokens until their total probability surpasses a set threshold, denoted as  $p$  (Holtzman et al., 2019).

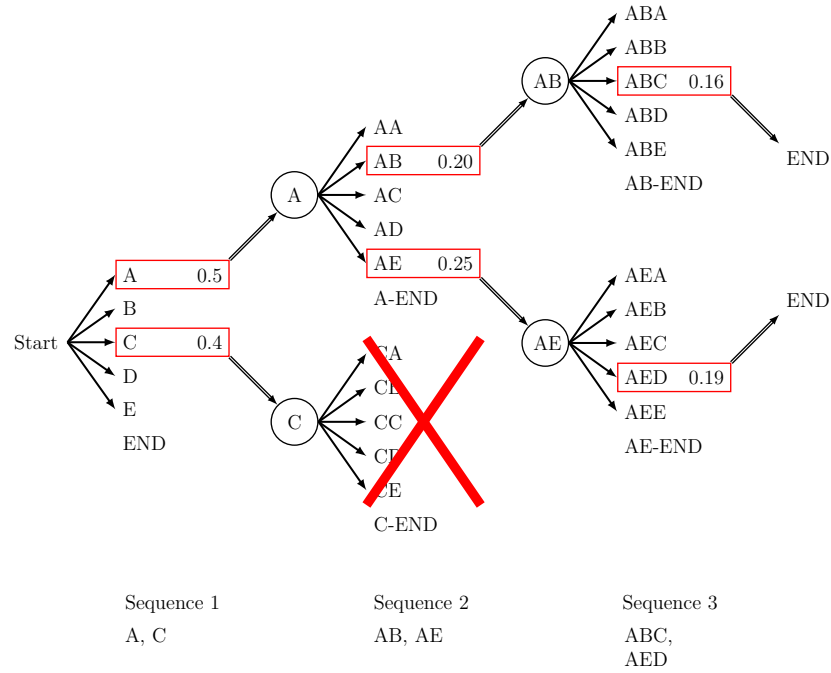
By combining Top- $p$  with Top- $k$ , it is possible to create a dynamic token selection



method that ignores less important words while maintaining adaptability in the selection process (von Platen, 2020).

**2.4.4 Beam Search.** Utilizing Beam Search, a well-established decoding strategy for text generation may help overcome the issue of repetitive tokens in the output sequence (Fig. 6).

**Figure 6**  
*Schema of Beam Search*



*Note: Adapted from: Foundations of NLP Explained Visually: Beam Search, How It Works, by Doshi, K., 2021.*

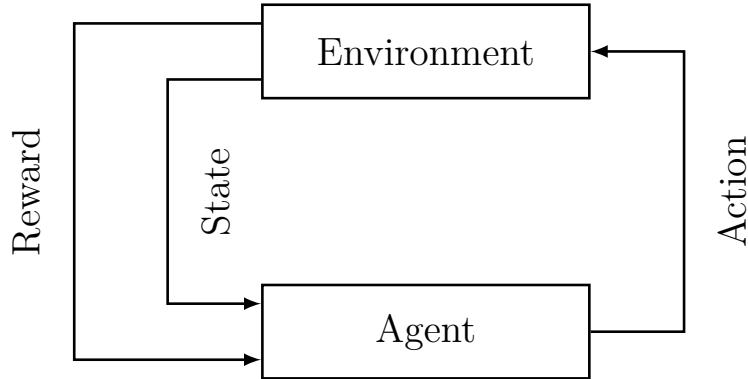
A beam is a set of candidate sequences that the algorithm maintains during the search process. Following, the beam search maintains a record of the most likely  $b$  next tokens, where  $b$  is denoted as the count of beams. Instead of selecting the best possible token at this time point, the next set of beams is determined by evaluating all potential next-token extensions of the current set and choosing the  $b$  most probable extensions. This process iterates until the end condition is encountered. Ultimately, the most likely sequence is chosen by ranking the  $b$  beams based on a scoring function, e.g. log probabilities (Tunstall et al., 2022, p. 24).

## 2.5 Reinforcement Learning

Besides Supervised and Unsupervised Machine Learning, Reinforcement Learning (RL) stands as a significant branch within the realm of ML. The main idea is to maximize a numerical reward function, that maps situations to actions. Unlike learning patterns, the learner (agent) focuses on identifying actions that yield the greatest increase in the reward function. To reach the biggest reward, the reinforcement learning agent has to promote the actions, that have been effective in producing the reward in the past. In the discovery of such actions, the agent selects actions, that have not been performed before. So, the agent uses its knowledge to obtain the best reward and parallel has to explore the reward to improve future selections. Another important feature of this kind of ML is the consideration of the whole problem of a goal-oriented agent reacting in an uncertain environment. RL consists of four main elements: policy, reward function, value function, and the model of the environment (Sutton & Barto, n.d.). Amiri et al. (2018) presented a simple schema as shown in Fig. 7.

**Figure 7**

*Schema of Reinforcement Learning*



*Note: Adapted from: A Machine Learning Approach for Power Allocation in HetNets Considering QoS, by Amiri et al, 2018.*

A policy is a description of how the agent behaves at a given time point. For the sake of simplification, a policy serves as a guiding mechanism connecting the observations within the environment to the actions that need to be undertaken in response to those observations. The reward function establishes the primary objective. It associates a numerical value, known as a reward, with every observed state in the environment, indicating how appealing that state is. The fundamental goal of a reinforcement learning agent is to optimize its cumulative reward over time. This reward function essentially defines which events are preferred by the agent. A value function outlines what is considered advantageous in the grand scheme of things. The value of a state represents

the overall sum of rewards an agent can anticipate gathering in the future, commencing from that particular state. An environment model imitates how the environment works. For instance, it predicts what will happen next from a given state and action, including the next state and the reward. These models are used to support decisions by considering possible future situations before experiencing them (Sutton & Barto, n.d.).

This approach can also be applied to NLG tasks with LLM. A method called PPO, introduced by OpenAI researchers (Schulman et al., 2017), is one way to do this. PPO maintains a balance by collecting data through interactions with the environment and improving an objective function through stochastic gradient ascent (Schulman et al., 2017).

### 3 Research Design and Data

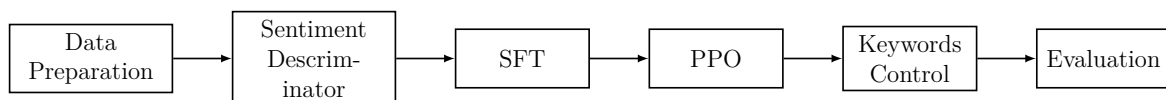
In this chapter, methods used for CTG with sentiment token and keywords are depicted. Additionally, it presents proposed adjustments.

The pipeline is presented in Section 3.1 and gives an overview of the steps that are proposed for the control process. The dataset containing reviews and texts written in German is introduced in Sect. 3.2. A crucial aspect of the pipeline’s success is its evaluation. Section 3.3 explains the metrics used to evaluate the performance of the pipeline and its components. The sentiment discriminator is introduced in Sect. 3.4. It is an important element since it evaluates the text’s sentiment and is used for the reward calculation in the optimization process with RL. In Sect. 3.5 and Sect. 3.6 the Fine-tuning and PPO used for control of sentiment-based text generation are explained. Section 3.7 depicts the way of forcing keywords in the generated text. The last section, Sect. 3.8, presents the way of the human evaluation of the text.

#### 3.1 Workflow

Figure 8 shows the workflow of the proposed approach. The first step contains data cleaning. During this stage, various cleaning approaches have been devised and im-

**Figure 8**  
*Workflow of this Thesis*



*Note: Own Creation.*

plemented to facilitate subsequent steps of the workflow. Following, various sentiment discriminators based on the BERT model have been established to find the best one. Parallel, the GPT-2 model has been fine-tuned on the data transformed into instruction indicating instructions. The sentiment discriminator is used in the RL process with PPO to improve the sentiment generation of the fine-tuned model. As the next step, the trained model is used to generate text containing specified keywords by logit modification and utilizing different decoding strategies. Lastly, the quality is examined with human evaluation through a survey.

## 3.2 Data

The data from Guhr et al. (2020a) is employed in this work research.

**3.2.1 Dataset Description.** Guhr et al. (2020b) proposed a sentiment discriminator for German text. They collected a new data corpus for German sentiment analysis. The following segments of their dataset (Guhr et al., 2020a) are utilized within this thesis: hotel reviews scraped from holidaycheck.de, movie reviews scraped from filmstarts.de, and emotions data. The summary of these datasets is contained in Tab. 1.

**Table 1**

*Results of the Analysis of Datasets.*

Dataset	No. reviews	Positives	Negatives	Neutral	Scale
Filmstars	71,126	40,015	15,611	15,500	0.0 - 5.0
Holidaycheck	4,831,290	3,995,507	388,741	447,042	1.0 - 6.0
emotions	1,306	188	1,090	28	Categorical

The Filmstars dataset comprises 71,126 user reviews in German, each of them labeled for sentiment. These reviews were gathered using a web crawler in October 2018. The dataset exhibits a slight class imbalance, with the majority of samples (40,015) being positive, while 15,611 are negative and 15,500 neutral. The reviews lay between 0.0 and 5.0. One of the positive reviews is cited below. It is contained in the file *filmstarts.tsv* in row 16.

*Monumental, episch und unerreicht. Peter Jackson gelingt die Quadratur des Kreises, die perfekte Umsetzung einer "unverfilmbaren" Vorlage. Die Trilogie ist bis zum heutigen Tag das gewaltigste Filmprojekt aller Zeiten und genießt zu Recht eine konkurrenzlose Alleinstellung im cineastischen Kosmos. Es ist die schamlose, authentische*

*Entführung des Zuschauers in eine andere Welt. Willkommen in Mitteleerde.*

The second dataset, Holidaycheck, features user reviews of diverse hotels, all written in German. The dataset encompasses a total of 4,831,290 data points, with 3,995,507 being positive reviews, 388,741 being negative reviews and 447,042 neutral. The data was sourced in 2018. The reviews lay between 1.0 and 6.0. An example of a positive review can be seen below. The source of this review is file *holidaycheck.indien.tsv*, row 23.

*Gute Lage am Strand! Tolle Lage direkt am Strand, wo am Abend das Leben nur so brummt und eine große Auswahl an Streetfood Ständen zu fairen Preise allerlei Köstlichkeiten bieten. Aktuell wird umgebaut, daher kann es tagsüber zu etwas Lärm kommen, insgesamt hat dies jedoch nicht gestört.*

Following a similar approach as Guhr et al. (2020b), the reviews within these datasets are categorized into two classes. Reviews with ratings below three are labeled as "negative" while those with ratings higher than three are classified as "positive". Ratings with a value of three points are considered neutral and will not be included in the analysis.

The *emotions.txt* file is relatively small, consisting of 1,306 samples. However, the majority of these samples (1,090 of them) are categorized as negative. This dataset primarily comprises a list of utterances that were recorded by the authors. Mostly, the data points contain only a few words. The sentiment label is contained at the beginning of the string and indicates positive, negative, or neutral sentiment. One of the positive sentences is shown as follows and can be found in line 1,092 (Guhr et al., 2020a).

\_\_\_label\_\_\_positive das ist echt schön

**3.2.2 Data Cleaning.** Data preparation is an essential step in NLP. First, the datasets mentioned in section 3.2.1 are combined into one to obtain a broad knowledge of data in the three topics: reviews of hotels, films, and different sentences depicting emotions. The data pre-processing can remove the noise from data, which can be contained in punctuation or stop words. The aim of pre-processing is the improvement of the accuracy of the ML model (De Silva, 2023).

The way of cleaning the data can affect the performance of the models. Due to this, different methods in different configurations are investigated, such as removing punctuation, stop words and lowercasing. Further, texts are cleaned to ensure consistency.

As the models used across proposed approaches are built on transformers architecture, it may not require extensive pre-processing, but tokenization, mapping tokens into integers, and padding or truncation (Lokare, 2023). The Transformer models are trained on the large text corpora, the fine-tuning process does not require a large amount of data, due to this only a subset of the dataset will be used for the model development.

One important feature of the data is the balance of the target classes in the training process. There is a risk of bias in the model toward the majority class. To address this problem, sampling can be used on the training dataset (Stewart, 2023).

### 3.3 Automated Evaluation Metrics

This section introduces different evaluation methods used in the frame of this work.

**3.3.1 Classification Metrics.** To evaluate the performance of the classification model (e.g. sentiment discriminator), accuracy, precision, recall, and F1-score as described in Vujovic (2021) are used. These metrics are based on a confusion matrix (Fig. 9) and its elements:

**Figure 9**  
*Confusion Matrix*

True Labels	Negative [0]	True negative	False positive
	Positive [1]	False negative	True positive
		Negative [0]	Positive [1]
		Predicted Labels	

*Note: Own Creation Based on: Classification Model Evaluation Metrics, by Vujovic (2021)*

- TP (True Positive) - Cases, where the model correctly predicts the positive class.
- FP (False Positive) - Cases, where the model incorrectly predicts the positive class when the true output is negative.
- FN (False Negative) - Cases, where the model incorrectly predicts the negative class when the true output is positive

- TN - (True Negative) - Cases, where the model correctly predicts the negative class.

Accuracy represents the ratio of correct predictions of the model to the total number of predictions. It is represented by the Eq. 8.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

Precision indicates the correctness of positive predictions, while recall signifies the completeness of positive predictions. The equations are following:

$$precision = \frac{TP}{TP + FP} \quad (9)$$

$$recall = \frac{TP}{TP + FN} \quad (10)$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (11)$$

In the scope of this thesis, the implementation of these metrics from *Scikit-learn* library (Pedregosa et al., 2011) are used.

**3.3.2 Perplexity.** As an automated metric of fluency, Perplexity will be utilized. In the context of NLG the Perplexity is used as a word-level method to evaluate a language probability model (Zhang et al., 2022). It states how sure the model is about its prediction.

$$PPL = \sqrt[n]{\prod_{i=1}^n \frac{1}{p(w_i|w_1w_2...w_{i-1})}} \quad (12)$$

Similarly to Pascual et al. (2020), the Perplexity of a text will be calculated utilizing a different model, than the one that generated it. In terms of LLMs, the Perplexity can be calculated also as

$$PPL = e^{loss} \quad (13)$$

as shown in (Hugging Face, n.d.-b). In the scope of this thesis, the loss is calculated with *distilbert-base-german-cased* (Gugger, 2020). The model receives inputs and labels of the tokenized text and produces an average negative log-likelihood for each token that is returned as the loss. Finally, the Perplexity is calculated as the exponential function applied to the loss term.

**3.3.3 SLOR.** Another score that can be used to evaluate fluency is SLOR (Kann et al., 2018). The formula assigns the score to sentence  $S$ . The score is based on the log-probability under an LM and it is normalized by the unigram log-probability and sentence length as below:

$$SLOR(S) = \frac{1}{|S|}(\ln(p_M(S)) - \ln(p_u(S))) \quad (14)$$

$p_M$  describes the probability assigned to the sentence under the given LM and  $p_u$  the unigram probability of the sentence (Kann et al., 2018). The probabilities are obtained with the support of model *dbmdz/bert-base-german-cased* (Chaumond, 2020).

**3.3.4 Flesch Reading Ease.** Assessing text fluency involves gauging readability as one of its key aspects. It measures how easily a reader can comprehend the text. One established method for the evaluation is the Flesch Reading Ease Score (FRE) proposed in Flesch (1948). It evaluates the approximate reading grade, ranging between 0 and 100. The higher the score, the more understandable the text becomes (Pecánek, 2021). The equation looks as follows (Pecánek, 2021):

$$FRE = 206.835 - 1.015 \left( \frac{\text{total words}}{\text{total sentences}} \right) - 84.6 \left( \frac{\text{total syllables}}{\text{total words}} \right) \quad (15)$$

This equation is also implemented into the Python library *textstat* and works for text written in German (“textstat”, n.d.).

**3.3.5 Coherence Score.** Coherence can be understood as a measure of the logical connections in a text. To evaluate it, the coherence score coming from topic modeling can be used. It can be calculated by utilizing the coherence framework proposed by Röder et al. (2015). This framework consists of four steps: segmentation, probability calculation, confirmation measure, and aggregation.

During the segmentation of a group of words, coherence indicates how strongly one part is supported by another. When dividing a word set, this process results in several pairs of subsets. The method used for probability estimation determines how probabilities are calculated from the original data source. Later, a confirmation measure evaluates the strength of how a set of words is supported by another set using their respective probabilities. Lastly, all confirmations from subset pairs are brought together into a single coherence score (Röder et al., 2015).

This method is implemented by Rehurek and Sojka (2011) as *CoherenceModel* in *Gensim* library. However, if the topics are not specified, a pre-trained model should



be used. In the scope of this master’s thesis, a *LDAModel* from *Gensim* is utilized, similarly as in Kapadia (2019).

Latent Dirichlet Allocation (LDA) is a topic modeling technique established by Blei et al. (2003). This is a generative probabilistic and a three-level hierarchical Bayesian model, that operates on the discrete data, among other text corpora. The main concept is that documents are made up of different mixes of hidden topics. Each topic is defined by a set of words that are often found together in documents.

**3.3.6 Success Rate.** Pascual et al. (2020) proposed a metric called Success Rate. This metric calculates the number of keywords ( $n_{key}$ ) that occur in the text divided by the total number of specified keywords ( $n_{total}$ ) as in Eq. 16.

$$SuccessRate = \frac{n_{key}}{n_{total}} \quad (16)$$

## 3.4 Sentiment Classification

Every text encompasses more than just factual information; it also comprises contextual elements. One such significant contextual aspect is sentiment. To automatically classify text as positive or negative, the sentiment discriminator uses NLP techniques and ML algorithms to analyze the semantic and contextual information within the text, ultimately assigning sentiment labels based on the sentiment expressed in the content.

Guhr et al. (2020b) prepared their sentiment discriminators based on FastText and BERT to distinguish between positive and negative sentiment. Also Samuels and Mcgonical (2020) created a sentiment discriminator to predict the sentiment of news. They used WordNet to calculate the sentiment score of sentences and phrases to predict the sentiment score of the whole text.

In this master’s thesis, the sentiment discriminator is used to distinguish between positive and negative sentiments of the given respective text. In this part, two methods are proposed. As the task of a sentiment discriminator is to classify text, the performance of the models will be evaluated with the following metrics: accuracy, precision, recall, and F1-score. The best-performing discriminator will be used in the optimization with RL. It will also serve as an evaluator of sentiment quality for texts generated in the scope of this thesis.

### 3.5 Fine-Tuning

To boost the performance of LLM for downstream tasks, Fine-Tuning on domain-specific data is beneficial. The process relies on taking a smaller dataset and training the general LLM on it. This refines the model’s capabilities and improves performance in a particular task or domain (SuperAnnotate, 2023).

In SFT, the LLM is adapted using a labeled dataset through supervised learning techniques. This process involves adjusting the model’s weights to optimize its performance on the specific task. SFT enables the model to grasp task-specific patterns and intricacies in the labeled data. The model acquires specialization through parameter adjustments tailored to the unique data distribution and task demands, leading to improved performance on the target task (Martinez, 2023).

In this stage of the master’s thesis, an AR model, namely German GPT-2, will be fine-tuned on the cleaned dataset with sentiment-specific instruction. Hugging Face (Hugging Face, n.d.-a) provides the employed GPT-2 model that was created by Minixhofer et al. (2022). Methods from the *TRL* library (von Werra et al., 2020) are used during the Fine-Tuning process. This library contains a broad set of tools for training Transformer models, also in SFT fashion.

The models will undergo evaluation using Perplexity to gauge fluency. Furthermore, coherence scores will be computed. Sentiment agreement (predicted vs. actual) will be assessed using classification metrics. Outside the confines of this master’s thesis, SLOR and FRE will be employed to enrich the model’s comprehension.

### 3.6 Proximal Policy Optimization

PPO (Schulman et al., 2017) has already been SOTA in RL. Schulman et al. (2017) introduce two variants: PPO Penalty and PPO Clip. The PPO Penalty uses KL-divergence as a soft penalty, which is scaled by parameter  $\beta$ . The update schema  $\Delta\theta^*$  is expressed by the formula

$$\Delta\theta^* = \arg \max_{\Delta\theta} \mathcal{L}_{\theta+\Delta\theta}(\theta + \Delta\theta) - \beta(\mathcal{D}_{KL}(\pi_{\theta}||\pi_{\theta+\Delta\theta})) \quad (17)$$

where the first part is the expected advantage of an update with  $\mathcal{L}$  being the surrogate advantage of updated policy  $\pi_{\theta} + \Delta\theta$  using samples from policy  $\pi_{\theta}$  and the second part being the penalty term built on KL-divergence. The main idea is to maintain a balanced and controlled policy update process in the PPO algorithm. After every update, the PPO assesses the update’s size. If the actual difference between policies is 1.5 times

more than desired, PPO makes future updates smaller by doubling the penalty factor  $\beta$ . Conversely, if updates are too small, PPO halves  $\beta$ , allowing for larger updates and expanding the permissible range for policy adjustments (van Heeswijk, 2022).

The second variant, PPO with a clipped objective, restricts the range within the policy and can change instead of changing penalties over time. There is an incentive to maintain proximity to the current policy since updates beyond the clipping range do not contribute to the updating process (van Heeswijk, 2022). The objective function is depicted below:

$$\mathcal{L}_{\pi_{\theta}}^{CLIP}(\pi_{\theta_k}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[ \sum_{t=0}^T \left[ \min \left( \rho_t(\pi_{\theta}, \pi_{\theta_k}) A_t^{\pi_{\theta_k}}, \text{clip}(\rho_t(\pi_{\theta}, \pi_{\theta_k}), 1 - \epsilon, 1 + \epsilon) A_t^{\pi_{\theta_k}} \right) \right] \right] \quad (18)$$

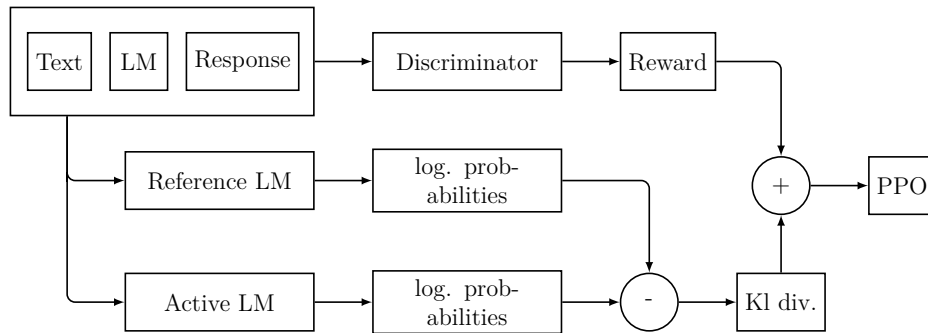
and  $\rho_t$  is the importance sampling ratio:

$$\rho_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \quad (19)$$

The expressions  $(1 - \epsilon) \cdot A$  and  $(1 + \epsilon) \cdot A$  do not depend on  $\theta$ . Consequently, the differentiation to  $\theta$  yields a gradient of 0. As a result, data points outside the trusted region are ignored, discouraging large updates (van Heeswijk, 2022).

Similar to SFT, the PPO algorithm for RL has been implemented in the *TRL* library (von Werra et al., 2020). The workflow, as suggested by von Werra et al. (2020), involves the utilization of two language models. The first model acts as a reference, while the second one serves as the actively trained model. The language model generates a response based on a query, which can serve as the start of a sentence. The schema is shown in Fig. 10.

**Figure 10**  
*Finetuning of LM Using Reinforcement Learning*



*Note: Adapted from TRL: Transformer reinforcement learning, by L. von Werra et al., 2020.*

The query and response are assessed using a reward function. The reward is a single

numerical output derived from a separate model. Lastly, query-response pairs are used to compute log probabilities for the tokens in the sequences. This is achieved using the actively trained model and a reference model, typically the pre-trained model before Fine-Tuning. The KL-divergence between both outputs acts as an extra reward signal to ensure that generated responses remain close to the reference language model. The active language model is then trained with PPO. This reinforcement learning process aims to refine the LM for generating sentiment-controlled text (von Werra et al., 2020).

The models prepared with SFT are trained for one and two epochs to investigate how the text generation metrics are performing and how a different number of training epochs affect the RL. The models will be evaluated with the Perplexity to measure fluency. Additionally, the coherence score will be calculated. The sentiment will be evaluated with classification metrics. Extending the original scope of this master’s thesis, SLOR and FRE will be used to enhance the understanding of the model.

### 3.7 Logits Modification Mechanism

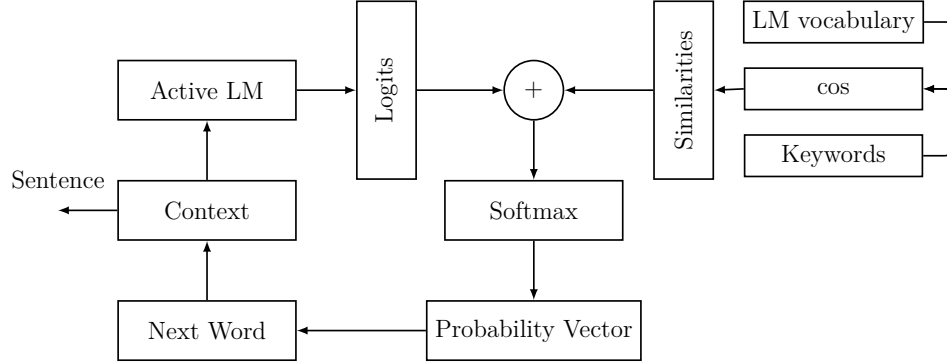
Pascual et al. (2020) split the text generation control into two groups. The first controls text generation with soft constraints, e.g. mood. This constraint has been utilized in this master thesis while generating sentiment-controlled text. The second kind is hard constraints. These are e.g. control with specific keywords.

Pascual et al. (2020) proposed a method Directed Beam Search, which controls the text generation with keywords. The key idea is to generate a new token by an LLM that is most similar to the keywords. Pascual et al. (2020) calculate the similarities between logits generated by the model and a keyword at each time step. He also uses stemming to check for the occurrence of the keywords. To transform the text to numerical representation he utilizes GloVe (Pennington et al., 2014). The calculated similarities are then squared and added to the model logits as in Eq. 20.

$$l'_i = l_i + \lambda \cdot \min(0, \cos(\gamma(t_i), \gamma(key_j)))^2 \quad (20)$$

The newly adapted logits are the sum of the logits  $l_i$  generated by the LLM and weighted by  $\lambda$  similarities between the  $i^{th}$  token  $t$  from the vocabulary of the LM and keyword  $key$ . The  $\gamma$  represents GloVe embeddings. The softmax function is applied to the adapted logits to convert them into probabilities. Subsequently, one token is sampled (Fig. 11). The generation process is repeated until all keywords are used in the text generation. This method compares keywords to text in the order of the keyword definition: the second keyword is taken into consideration when the first is

**Figure 11**  
*Mechanism of Logits Modification*



*Note: Adapted from Directed Beam Search: Plug-and-Play Lexically Constrained Language Generation, by Pascual et al., 2020.*

already used in the text generation. Beam Search is utilized as the decoding strategy.

During this work, the logit modification algorithm proposed by Pascual et al. (2020) is changed. Instead of taking keywords in order, it includes keywords independently from their order in the input list. Because FastText (FastText, n.d.) demonstrates superior performance compared to the GloVe algorithm, especially when dealing with rare or unknown words (Deep, 2020), it has been chosen for the creation of word embeddings in this master’s thesis.

In the frame of this master’s thesis, lemmatization is performed to ensure the agreement between the keyword and the newly generated word. Different sentiment-controlled models (created as in Section 3.6) are used as a generative model for this approach. Since the method is applied during the post-processing, different decoding strategies, such as Greedy Search, Top- $k$ , and Top- $p$  are tested. Also, a new approach for Beam Search with a keywords-forcing mechanism is proposed.

The generated text will be evaluated in terms of Perplexity, sentiment accuracy, and Success Rate.

### 3.8 Human Evaluation

There are different types of automated text evaluation, such as mentioned before Perplexity and SLOR scores. However, human evaluation enables a better understanding of the properties of the generated text. Some researchers such as Dathathri et al. (2019) evaluate the text generation by humans. van der Lee et al. (2021) explains in his guideline of best practices about human evaluation for NLG, that there are two types of evaluation: intrinsic and extrinsic.

Intrinsic approaches focus on assessing the inherent qualities of a system’s output, often by gathering judgments directly through methods like questionnaires to evaluate aspects like fluency. On the other hand, extrinsic or task-based approaches assess the system’s effectiveness by examining how well it accomplishes the primary task it was designed for (van der Lee et al., 2021).

A questionnaire is one of the easiest and cheapest ways to use the intrinsic approach. In the scope of this research, it will be used to evaluate the human perception of texts generated by models created in the scope of this thesis. The questionnaire is designed to evaluate the perception of texts generated using sentiment and keyword control in terms of sentiment, fluency, and coherence. To support a selection of the model two metrics are used: Perplexity and Success Rate. The model is chosen from a selection of optimized models employing various decoding strategies. This model is used to generate texts with different controls. Texts without keyword control rely only on Perplexity for selection, favoring those with the lowest Perplexity scores. For texts generated with additional keywords forcing, the Success Rate is utilized parallel to Perplexity. The winning texts have low Perplexity and a high Success Rate. As the aim of this survey is to evaluate the text generation performance only small adjustments in the texts are made manually.

An Inter-Annotators Agreement (IAA) is used to evaluate the consensus among multiple survey responders about the text quality expressed in the survey. It measures consistency among annotators (Artstein, 2017). In the scope of this work, the Multi- $\kappa$  (Davies & Fleiss, 1982), generalization of Cohen’s  $\kappa$ , is used.

## 4 Experiments

In this Chapter, the execution of the experiment is presented. After the combination of different datasets into one (Sect. 4.1), the sentiment discriminator is created using two different approaches based on the BERT model - with CNN layers and classification head. Then the German GPT-2 model is fine-tuned using SFT (Sect. 4.3.1) and then RL (Sect. 4.3.2) for sentiment control. After this optimization, the algorithm for forcing keywords into the generated text is introduced in Sect. 4.4. In the final section (Sect. 4.5), the design of the survey for evaluation of the quality of the generated texts is presented. The experiments are conducted in Google Colab Pro using Python as the programming language.

## 4.1 Data Combination and Preprocessing

First, the data of the emotion dataset is prepared. The label is contained in the sequence as the first word, where the sentiment was contained, e.g. *\_\_\_label\_\_\_negative*. For this, the sentiment needed to be extracted. Similarly as (Gühr et al., 2020b) the hotel and movie reviews rated below 3 are classified as negative, larger than 3 as positive, and equal to 3 are neutral. From all datasets, the positive and negative reviews are selected and combined into one dataset.

Different data pre-processing approaches are applied. A more detailed description of the applied cleaning procedures is presented along with the model. Generally, a random selection of samples is drawn from the combined dataset, and these samples are then organized into a *pandas* data frame (McKinney, 2010; pandas development team, 2020). The pre-processing is performed using the methods provided in the libraries: *NLTK* (Bird et al., 2009), and *spaCy* (Honnibal & Montani, 2017). All datasets underwent cleaning procedures such as replacing special German characters, e.g. *ae* with *ä*, and removing elements such as emojis, phone numbers, and multiple consecutive empty spaces. These steps are taken to ensure the clarity and consistency of the text data. The split into train, validation, and test dataset is performed with *Scikit-learn* (Pedregosa et al., 2011). Resampling of the training datasets is carried out due to an observed imbalance in sentiment classes within the dataset, with a predominant bias towards positive sentiment. Methods from *Imbalanced-learn* library (Lemaître et al., 2017) are used to deal with the class imbalance.

## 4.2 Sentiment Discriminator

In this section, different sentiment discriminators are created and evaluated. Their task is to classify specific texts as positive or negative.

**4.2.1 Data Processing.** Three datasets used for the training of sentiment discriminators are prepared in the following ways:

1. All words are lowercased, and punctuation and stop words are removed.
2. All words are lowercased.
3. The capitalization, punctuation, and stop words are contained.

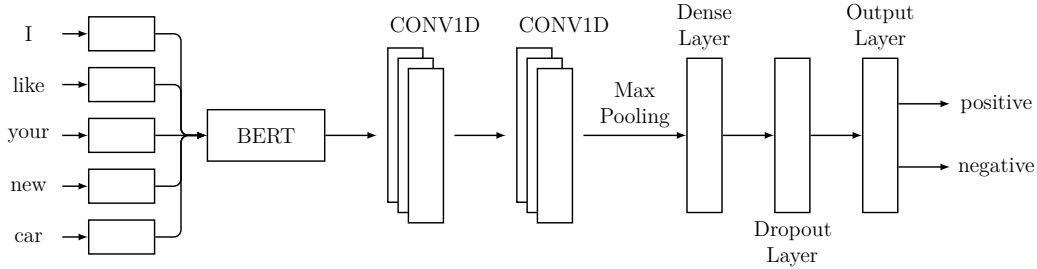
In the first step, a random selection of 50,000 samples is drawn from the combined dataset. Split into training, validation, and test datasets is performed. The training

datasets for the models described in section 4.2.2 are undersampled and oversampled in section 4.2.3.

**4.2.2 BERT model with Convolutional Neural Network.** In this section, three models with identical architecture are trained using datasets from section 4.2.1 and implemented using the *TensorFlow* library (Martín Abadi et al., 2015). The approach involves leveraging a pre-trained BERT (*dbmdz/bert-base-german-cased* created by Chaumond (2020)) with frozen layers. This model translates the input text into embeddings which are then processed by a Convolutional Neural Network (CNN) to extract relevant features. Two CNN layers, each with filters of size 50 and padding set to two, are constructed. ReLU serves as the activation function in these CNN layers. Subsequently, Global Max Pooling Layers are used to reduce the resolution of features. This helps by increasing the CNN’s robustness to noise (Dang et al., 2020). Following, a Dense layer comprising 256 neurons, activated by ReLU and employing kernel normalization, is integrated. Additionally, a Dropout layer with a dropout rate of 0.2 is incorporated. Adam optimizer with a learning rate of  $1 \cdot 10^{-4}$  is used. The output layer consists of two neurons to generate logits for both sentiment classes (positive or negative). The simplified model is depicted in Fig. 12.

**Figure 12**

*Sentiment Discriminator Built on BERT with CNN Layers*



*Note: Own Creation.*

To prevent overfitting, the models employ early stopping with a patience level set to six based on the validation loss. Training spans 25 epochs, employing a batch size of 128 during the training process. Every model predicts classes by first generating logits, and then converting these scores into probabilities using the softmax function. Finally, it selects the class with the highest probability as the predicted class for each example in the test dataset. In later work, the models are referenced as:

- Discriminator 1: Model trained on the dataset, where all words are lowercased, and punctuation and stop words are removed.



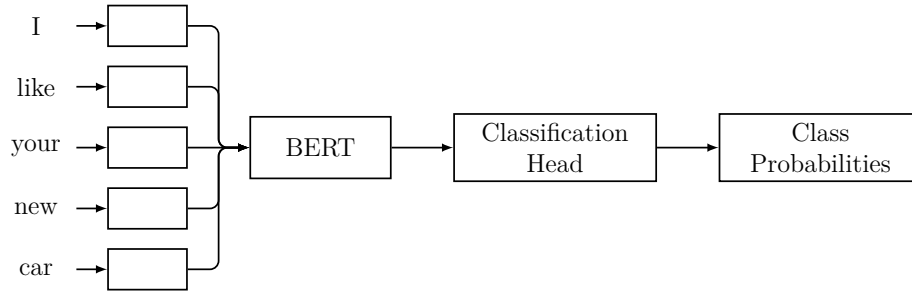
- Discriminator 2: Model trained on the dataset, where all words are lowercased.
- Discriminator 3: Model trained on the dataset, where the capitalization, punctuation, and stop words are contained

Section 5.1 presents the results of the evaluation of these sentiment discriminators.

**4.2.3 BERT model with Classification Head.** Another approach to harness the capabilities of LLM in classification algorithms is through the utilization of a classification head. The classification head can process the sequence of output tokens generated by the transformer model (e.g. BERT) and translate them into predictions for one or multiple classes (Manocchio et al., 2023). The schema of the model can be seen in Fig. 13.

**Figure 13**

*Sentiment Discriminator Built on BERT with Classification Head.*



*Note: Own Creation.*

During this phase, the sentiment discriminator (referenced later as Discriminator 4) is created on the minimally cleaned data that contains capitalization, punctuation, and stop words. The training’s concept is based on the idea presented by Hugging Face (Hugging Face, n.d.-c). The text data undergoes tokenization in batches to prepare it for model training, validation, and testing. During the tokenization truncation to the maximal input length is performed in order to convert text of varying length into fixed-size tensors.

The data collator is responsible for combining and processing individual training examples into batches suitable for training a model. In this case, the data collator prepares tokenized sequences by ensuring uniform length through padding.

The German BERT is initialized with the classification head, which is implemented within the class *BertForSequenceClassification* in the *transformers* library (Wolf et al., 2020). The labels are converted to sentiment and vice versa since the model needs to translate classes properly during the training and evaluation phase, as shown in Hugging Face (n.d.-c).

The model is trained with specific parameters. The learning rate is set to  $2 \cdot 10^{-5}$ . 16 samples per batch are processed at once during both training and evaluation. The training is performed for one epoch. Additionally, the model's performance is evaluated every epoch. The regularization parameter, weight decay, is set to 0.01.

Similarly as in Subsection 4.2.2, the predicted class is the one with the highest logit.

## 4.3 Sentiment Control

In this section, the approach for the creation of sentiment-based text generation is presented. First, the GPT-2 model undergoes Fine-Tuning in the SFT fashion. Later, it is optimized with RL to improve the text generation toward a given sentiment the RL approach is used.

**4.3.1 Supervised Fine-Tuning.** The German GPT-2 model (Minixhofer et al., 2022), might face challenges in adapting to new, specific domains and tasks due to its training on a broad text corpus. To overcome this limitation, employing an SFT approach can enhance its performance in the desired domain and task.

A random selection of 500,000 samples is drawn from the combined dataset. The model undergoes Fine-Tuning using a SFT approach, following a set of instructions derived from a minimally cleaned dataset that retains capitalization, punctuation, and stop words. After resampling the training dataset consists of 73,606 samples. The function for instruction creation is defined in Listing 1. Its output has a form: *[sentiment token] text*.

### Code Listing 1

*Creation of Instructions for SFT. Note: Own Creation.*

```

1 def create_instruction(sent,text):
2     '''
3     create instructions that will be used for supervised fine-tuning.
4     As an instruction we create text in the form "[sentiment] text".
5     Finally, we take the first 512 words as the instruction.
6     '''
7
8     text=f"[{sent}] {text}"
9     words = text.split()
10    text = ' '. join(words[:512])
11    return text
12
13 data["instructions"]=data.apply(lambda row: create_instruction(row['
    sentiment'], row['preprocessed_text']), axis=1)

```

The column *sentiment* is the label of the review (either "*positive*" or "*negative*") and the column *preprocessed\_text* is the cleaned review. To reduce the size of reviews, the first 512 words are selected.

The methods for Fine-Tuning have been implemented into the *TRL* (von Werra et al., 2020) library from *Hugging Face*. The model is fine-tuned on the instructions dataset with a learning rate of  $5 \cdot 10^{-5}$  using the class *SFTTrainer* (Listing 2). The maximal sequence is set to 1024.

## Code Listing 2

*SFT Training. Note: Own Creation.*

```

1 config=TrainingArguments(
2     output_dir="model_output",
3     learning_rate=5e-5,
4     num_train_epochs=3###change to 1,2,3
5 )
6
7 trainer = SFTTrainer(
8     model,
9     train_dataset=dataset,
10    dataset_text_field="instructions",
11    max_seq_length=1024,
12    args=config
13 )

```

Three models are created in this process:

- SFT1: Model trained in an SFT manner for one epoch.
- SFT2: Model trained in an SFT manner for two epochs.
- SFT3: Model trained in an SFT manner for three epochs.

During this phase, the author was faced with a significant obstacle; the primarily selected, well-established German GPT *dbmdz/german-gpt2* model (Schweter, 2020) did not work with SFT. Due to this the model was changed to *benjamin/gpt2-wechsel-german* (Minixhofer et al., 2022). Subsection 5.2.1 presents the results of the evaluation of the fine-tuned models.

**4.3.2 Reinforcement Learning Optimization.** For the sentiment optimization the possibilities of the *TRL* library (von Werra et al., 2020) are used as shown by von Werra (2023). The same dataset is taken for the sentiment optimization with RL as for the SFT process and is organized into dataset structure from the *Datasets* library

(Lhoest et al., 2021). Models fine-tuned in Section 4.3.1 are optimized with the PPO algorithm. They are optimized using mini-batches that are 16 examples with 51,200 steps. The learning rate for the training is set to  $1.41 \cdot 10^{-5}$ .

### Code Listing 3

*PPO Model Initialization. Note: Own Creation.*

```
1 model = AutoModelForCausalLMWithValueHead.from_pretrained(config.  
    model_name)  
2 model_ref = create_reference_model(model)  
3  
4 tokenizer = AutoTokenizer.from_pretrained(config.model_name)  
5 tokenizer.pad_token = tokenizer.eos_token  
6  
7 ppo_trainer = PPOTrainer(config, model, model_ref, tokenizer, dataset,  
    data_collator=collator )
```

Two models are initialized (Listing 3) from the SFT model. The first one is the reference model and the other one is actively optimized. Tokenization breaks down raw text into smaller, meaningful units called tokens, facilitating analysis and processing. As in the sequence processing tasks, sequences usually have varying lengths. To uniform them, padding tokens are added to shorter sequences to match the length of the longest sequence. The last line in Listing 3 essentially makes the tokenizer use the end-of-sequence token as the padding token.

In addition, a reward function for two distinct tasks must be created: 1) for generating text with a positive sentiment and 2) for generating text with a negative sentiment. As the reward function is calculated based on logits created by the sentiment discriminator (Discriminator 4, Section 4.2.3), there is a need to establish the pipeline for this model and create functions, that calculate the rewards based on it. Thus, the sentiment discriminator is initialized using the pipeline from the *transformers* library (Listing 4).

### Code Listing 4

*Calculation of Model's Reward. Note: Own Creation.*

```
1 classifier = pipeline("sentiment-analysis", model="/content/drive/MyDrive  
    /Masterthesis/Models/sentiment_discriminator_bert_finetuned",**  
    sentiment_pipe_kwargs)  
2  
3 def get_logits(texts):  
4  
5     scores_texts=[]  
6     for text in texts:  
7         output=classifier(text)[0]  
8         score_dict = {item['label']: item['score'] for item in output}
```

```

9   negative_score = score_dict.get('NEGATIVE', 0.0)
10  positive_score = score_dict.get('POSITIVE', 0.0)
11  # Create a list with negative score first, then positive score
12  scores = [negative_score, positive_score]
13  scores_texts.append(scores)
14
15  return scores_texts
16
17 def logit_to_reward(logit, task):
18
19     scores=[]
20     for i in range(len(logit)):
21         if task[i] == "[negative]":
22             scores.append(logit[i][0])
23         elif task[i] == "[positive]":
24             scores.append(logit[i][1])
25
26     return [torch.tensor(score, dtype=torch.float32) for score in scores]

```

During the optimization process, texts are generated in mini-batches (based on input queries, that consist of randomly selected sentiment tokens and text beginnings (five first words from the original texts)). For a given text, the function *get\_logits* (Listing 4) employs the sentiment discriminator and returns two logits (each one for positive and negative class). Then, the function *logits\_to\_reward* decides which logit is used as a score based on the task. If the task is to create a positive sentence, the logit from the positive class is used, and vice versa. The score is further used as a reward. The reward functions for positive and negative tasks are monitored with *wight&bias* (Biewald, 2020). The generated rewards are passed along input query and response tensors as feedback to the PPO trainer. This rewards/penalizes the undertaken action and reinforces future text generation. This aims to improve sentiment generation without any extra risk of overfitting.

In the scope of this research, two models after SFT from section 4.3.1 are fine-tuned using RL: SFT1 and SFT2. Both models are trained using RL for one and two epochs to investigate, whether the number of epochs influences the text generation. This resulted in the creation of four new models:

- SFT1RL1: The SFT1 model fine-tuned with RL for one epoch,
- SFT1RL2: The SFT1 model fine-tuned with RL for two epochs,
- SFT2RL1: The SFT2 model fine-tuned with RL for one epoch,

- SFT2RL2: The SFT2 model fine-tuned with RL for two epochs.

Section 5.2.2 presents the results of the evaluation of the optimized models.

## 4.4 Keyword Control

As the performance in sentiment control is satisfying, keyword control is added. In this section, the mechanism of logit modification and its interaction with different decoding strategies is presented.

**4.4.1 Logit Modification Mechanism.** The controlled mechanism is done based on the method proposed by Pascual et al. (2020). In the scope of this thesis, instead of selecting keywords in the order they are given in the keywords set, they are forced into the text independently and extended to different decoding strategies.

In the first step, the lemmatization method is established. After numerous trials with different methods from *NLTK* (Bird et al., 2009) and *SciPy* (Virtanen et al., 2020) libraries, *Tree Tagge* (Schmid, 1999, 2013) is selected due to the unsatisfying performance in German of the two first libraries. This tool annotates text with part-of-speech and lemma information (Schmid, n.d.). It is worth noticing, that the output string of this method required some cleaning, as it contains the generated lemmas and indication of its part of speech. Due to this, lemmas are extracted from the output string.

In the next step, the model and the associated tokenizer are initialized with classes *AutoTokenizer* and *AutoModelWithLMHead* from the *transformers* library from (Hugging Face, n.d.-a).

The vocabulary of the models (50,257 words) is converted to the embedding space; in this case, using FastText (FastText, n.d.) downloaded from the *huggingface\_hub* (Hugging Face, n.d.-a). The mechanism is as follows. The word’s indexes are decoded and the generated words are cleaned. The empty spaces at the beginning and the end of the word are removed. Then they are transformed with FastText to vectors with a dimension of 300.

Pascual et al. (2020) proposed two functions for the text generation. The first is responsible for text generation with guidance until all keywords are used in the text generation. If all keywords are already used in the generation process, the second function creates words without constraints. In Listing 5 in Appendix A, the body of the adapted function for text generation with guidance is shown. In contrast to the function proposed by Pascual et al. (2020), the order of keywords in the keyword set does not matter.

The adapted function for guidance works in the following way. The current context (text) is passed to the tokenizer for encoding. The resulting tensor is then used by the language model in the process of the next word prediction and the resulting logits are obtained. The similarities between a keyword and the model vocabulary are calculated. The similarities are squared and added to the model logits to penalize the dissimilar words in favor of the more similar ones. The newly calculated logits are used in the decoding process. The softmax function is employed to derive probabilities from the logits. Sampling from the multinomial distribution enables obtaining the index of the following word. This is repeated for every keyword. Then, the best-performing index (the one with the highest probability) obtained for all keywords is selected and the current context is decoded. The predicted word is transformed into a lemma and compared to the lemmas of keywords. If the keyword has already been utilized in the text generation, it will not be considered in the further generation. If all the keywords are used, the next words will be generated without keyword control. The code employs the functionalities of the *Torch* library (Collobert et al., 2011).

The mechanism for the prediction of the next word without guidance is similar but more straightforward. The model generates logits, that are transformed into probabilities. After decoding, one word is sampled from the multinomial probability and encoded (Pascual et al., 2020).

**4.4.2 Decoding Strategies.** In the scope of the thesis, four main decoding strategies are investigated: Greedy Search, Top- $k$ , Top- $p$ , and Beam Search. The first three methods correspond to the definition presented in Sect. 2.4. The Top- $k$  and Top- $p$  approaches follow the code provided by Pascual et al. (2020). The Greedy search is adapted from Top- $k$  as its special case where  $k = 1$ . The decoding strategies are implemented inside functions for text generation with and without guidance. This can be seen in the Listing 5 in Appendix A. In contrast, the proposed Beam Search is created outside the generation functions.

**Beam search.** The proposed Beam Search differs from the classical one, which is described in section 2.4.4. The design of a new beam search was required since the proposed by Pascual et al. (2020) used keywords’ order as in the input set. The code of the Beam Search created in this thesis is shown in Listing 6 (Appendix B).

At the beginning of each iteration, there is a generation of parents (e.g., the start of a sentence). During each iteration, the model generates  $b$  children for each parent using functions for a token generation with and without guidance. In this implementation, a parent is deleted in the moment, that its  $b$  children are generated. At the end of each

iteration, the children (usually  $b$  times the number of parents) will form the generation of parents for the next iteration. So the number of children is exponentially increasing, which dramatically increases the required computational resources and time. As an example, assuming  $b = 3$ , there are 3 children in the first generation, 9 in the second, 27, 81, 243 ... .

$$wPPL = \frac{PPL}{n_{key} + 1} \quad (21)$$

Therefore the process stops after a given number of iterations (e.g. 4) and takes the best child of that generation as a new (single) parent to relaunch the process until a specific number of words in the sequence is reached. For this, a new score is proposed: Weighted Perplexity (wPPL). This is expressed as in the Eq. 21, where  $n_{key}$  is the number of utilized keywords and  $PPL$  is the Perplexity of the generated sequence. The sequence with the lowest  $wPPL$  score is then selected as the best one. If all keywords are used, then the text is generated without keyword guidance. Ultimately, the outcome is a sentence striking a balance between simplicity and alignment with the provided keywords, differing in approach and termination conditions from the original beam search method.

## 4.5 Survey Design

The automated metrics can evaluate the generated text to some extent but may not capture all nuances of the text quality. For better understanding, a human evaluation supports the analysis.

A survey with 18 texts (A1 to F3 in Tab. 2) is created. To lower the load for every participant, the survey is divided into three paths (green, yellow, and violet). Each of them contains six texts.

For rows A and B, the users evaluate the mutual influence of the text sentiment and keywords sentiment. Examples of negative keywords are *"hässlich"*, *"langweilig"* and *"Lobby"*. *"schön"*, *"hässlich"* and *"Lobby"* are example of mixed keywords. The corresponding texts are generated with the model SFTnRLm with a selected decoding strategy, that performs best in terms of Perplexity and Success Rate while text generation with nouns. Every text is selected as the one with the best Success Rate and Perplexity.

Rows C and D rows are created to evaluate the influence of Fine-Tuning on sentiment control, comparing the performance of models GPT-2, one model after SFT, and one after RL. As for the model after RL evaluation, the same SFTnRLm model as for rows



**Table 2**  
*Survey Design.*

	1	2	3	
A	key: negative	key: positive	key: mixed	sent: positive
B	key: negative	key: positive	key: mixed	sent: negative
C	GPT-2	SFTn	SFTnRLm	sent: positive
D	GPT-2	SFTn	SFTnRLm	sent: negative
E	Greedy Search	Top- $k$	Top- $p$	sent: positive key: nouns
F	Greedy Search	Top- $k$	Top- $p$	sent: negative key: nouns

A and B without keyword control is employed. For the analysis of the sentiment performance of the SFT, the SFTn model is selected that is previously used for SFTnRLm. The texts are chosen based on the best Perplexity score.

Rows E and F are designed to investigate the influence of keywords (nouns only) on the quality of the text generation using selected decoding approaches: Greedy Search, Top- $k$ , and Top- $p$ . The texts are generated with the same SFTnRLm model with the same keywords (nouns) across all sentiments and decoding strategies. For the evaluation, the texts with the same keywords used and the smallest Perplexity are selected.

The survey is created in Google Forms and is conducted in German. At first, the demographics data (age and gender) is obtained (Fig. 20 in Appendix D). Then, every user selects a color, which leads to the specific path (green, yellow, violet). All texts used in the survey are generated with a sentiment token followed by "*Wir waren in diesem Hotel.*". The texts contained in the survey are not cleaned except for the removal of unfinished or repeated words/phrases at the end. Every survey's path includes six texts as shown in Tab. 2. The texts are presented in Appendix E. The users have to evaluate texts in three categories: sentiment, fluency, and coherence. Example of a text

evaluation is shown in Fig. 21 in Appendix D.

For every text, the same items (statements) are prepared for evaluation. The users' evaluations lay on a Likert-type scale from 1 to 5, where 1 means "*I strongly disagree*" and 5 "*I strongly agree*". The items (see Tab. 23 in Appendix F) are designed to allow the assumption that the stronger agreement indicates a better performance as more participants are convinced about its validity. The description, mean ranges and interpretation of Likert scale is presented in Tab. 22 in Appendix F. Different coefficients for IAA have been implemented in the *NLTK* Library (Bird et al., 2009). In the scope of this work, the function *multi\_kappa* is used to calculate IAA.

To evaluate properly the sentiment agreement score of the negative statements, the item scoring is reversed (Carifio & Perla, 2007). To ensure consistency in the analysis of the sentiment agreement, the score of the text with negative sentiment is re-evaluated by  $score = 5 - actual\ score + 1$ .

To evaluate the results, the statements for fluency and coherence (see Appendix F) are combined into two latent variables describing those two categories respectively. In the first step, the items for fluency and coherence are tested with the Cronbach's Alpha test from *penguin* library (Vallat, 2018) to measure the internal consistency among statements belonging to one category. In the next step, the new latent variables are created by calculating the mean among items of the respective category. For the final evaluation, the mean and standard deviation are calculated for sentiment agreement, fluency, and coherence for every evaluated text.

## 5 Results

The evaluation assesses the effectiveness and performance of the implemented methods, providing insights into their applicability and areas for potential improvement. First in Sec 5.1, the performance of the sentiment discriminator is presented. Additionally, the best-performing model in terms of sentiment prediction is selected. Section 5.2.1 outlines the evaluation of the Fine-Tuning for sentiment-controlled text generation task. Since the texts generated by SFT models match the selected sentiment only partially, the RL optimization is applied to improve the performance. Section 5.2.2 presents the evaluation of this process. Following Sect. 5.3 contains the evaluation of the performance of the models when adding keyword control. Lastly, Sect. 5.4 reveals how the users perceive generated texts.

## 5.1 Performance of Sentiment Discriminator

Various measures are employed to assess the performance of sentiment discriminators. Since the validation dataset is imbalanced, it is important to not focus only on the accuracy but also to evaluate the results with precision, recall, and F1-score.

**Table 3**

*Performance of Sentiment Discriminators.*

Model	Accuracy ( $\uparrow$ )	Precision ( $\uparrow$ )	Recall ( $\uparrow$ )	F1-Score( $\uparrow$ )
Discriminator 1	0.87	0.97	0.88	0.92
Discriminator 2	0.88	<u>0.98</u>	0.89	0.93
Discriminator 3	0.89	0.97	0.91	0.94
Discriminator 4	<u>0.95</u>	0.97	<u>0.97</u>	<u>0.97</u>

An initial observation can be made based on the analysis of evaluation metrics for models employing both BERT and CNN Layers (Discriminator 1-3) as depicted in Tab. 3. The Discriminator 1 is trained on the dataset that is extensively cleaned. It comes with the weakest accuracy, recall, and F1-scores. The Discriminator 2 is cleaned less extensively and performs better on the test set as it achieves higher results than the Discriminator 1. The best performance out of the three CNN-based models is achieved by the Discriminator 3 which is trained on minimally cleaned data. It can be concluded, that more extensive cleaning of the data worsens the performance of the models that are built with the BERT. Too extensive cleaning can potentially lead to information loss.

The fourth model in Tab. 3, Discriminator 4, is the BERT with a classification head trained on the minimal cleaned data. It outperforms the other sentiment discriminators in terms of all classification matrices. As the best-performing model, it is utilized in the RL process to calculate the reward score.

Figure 14 shows the evaluation of proposed models in confusion matrices. The number of False Negatives drops significantly from Discriminator 1 (Fig 14a; FN: 526) to Discriminator 4 (Fig. 14d; FN: 121), showcasing the refinement of the predictive performance.

## 5.2 Evaluation of Sentiment Control

This section presents the evaluation of the models fine-tuned in the SFT manner (Subsect. 4.3.1) and optimized with RL (Subsect. 4.3.2).

**Figure 14***Confusion Matrices for Discriminator 1-4***(a) Discriminator 1.**

True Labels	Negative [0]	339	132
	Positive [1]	526	4003
		Negative [0]	Positive [1]
		Predicted Labels	

**(b) Discriminator 2.**

True Labels	Negative [0]	368	103
	Positive [1]	492	4037
		Negative [0]	Positive [1]
		Predicted Labels	

**(c) Discriminator 3.**

True Labels	Negative [0]	361	110
	Positive [1]	417	4112
		Negative [0]	Positive [1]
		Predicted Labels	

**(d) Discriminator 4.**

True Labels	Negative [0]	350	121
	Positive [1]	121	4408
		Negative [0]	Positive [1]
		Predicted Labels	

*Note: Own Creation*

**5.2.1 Evaluation of Supervised Fine-Tuning.** The models are evaluated and compared to the original GPT-2 model. A prompt based on sentiment tokens and the first ten words from the original review is prepared. The evaluation is conducted based on 50 positive and 50 negative texts.

The quality of the generated text is a crucial consideration, often assessed through metrics like Perplexity, FRE, and SLOR (Appendix G, Listing 7). Text coherence, crucial to the overall quality, is evaluated through a coherence score (Appendix G, Listing 8). The accompanying Tab. 4 presents the evaluation outcomes. To assess sentiment quality, Discriminator 4 uses the sentiment token as the ground truth. The sentiment token is removed from texts for evaluation.

The impact of SFT on Perplexity seems negligible (Tab. 4). The models maintain

**Table 4***Quality of Texts Created with SFTn Models.*

Model	Perplexity ( $\downarrow$ )	SLOR ( $\uparrow$ )	FRE ( $\uparrow$ )	Coherence Score ( $\uparrow$ )
GPT-2	<u>1.12</u>	<u>1.11</u>	75.76	0.863
SFT1	<u>1.12</u>	1.05	<u>79.91</u>	0.863
SFT2	1.13	1.08	79.88	<u>0.864</u>
SFT3	<u>1.12</u>	1.10	78.61	<u>0.864</u>

**Table 5***Sentiment of Texts Created with SFTn Models.*

Model	Accuracy ( $\uparrow$ )	Precision ( $\uparrow$ )	Recall ( $\uparrow$ )	F1-score ( $\uparrow$ )
GPT-2	0.68	0.80	0.68	0.64
SFT1	0.84	0.88	0.84	0.84
SFT2	<u>0.87</u>	<u>0.90</u>	<u>0.87</u>	<u>0.87</u>
SFT3	0.82	0.86	0.82	0.82

their ability to generate text as fluently as the original GPT-2 model.

The GPT-2 model demonstrates a higher SLOR score compared to the fine-tuned models. This signifies that according to the evaluation, the text generated by GPT-2 is deemed more probable.

When analyzing the readability score calculated with the FRE, the fine-tuned models have higher scores than GPT-2 (see SFT1 and SFT3). This means that their texts are easier to understand than the ones generated by the GPT-2 model. The coherence score varies between 0.863 and 0.864, which indicates quite good coherence, but the difference between models is neglectable.

As the aim of the process is to enable control over the sentiment in the text generation process, the accuracy, precision, recall, and F1 - score of the texts are evaluated in Tab. 5. This is computed by the sentiment discriminator shown in Subsect. 4.2.3.

Upon one epoch of SFT, there is a significant increase in all sentiment metrics compared to GPT-2, indicating improved sentiment generation. However, after two epochs, there is only a slight increase in metrics, followed by a decrease upon reaching three epochs of training.

Texts generated by the proposed models match the sentiment given as a token better than the ones generated by the original GPT-2 model. Consequently, Fine-Tuning the model for just one epoch appears sufficient since further training does not notably enhance sentiment generation.

**5.2.2 Results of Reinforcement Learning.** Similarly to the evaluation of the models trained with SFT, the same metrics are employed. The texts are generated based on 50 positive and 50 negative sentiment tokens. The input prompt also contains the first 10 tokens from the original text and its sentiment as a token. As for the quality of the text, Perplexity, FRE, and SLOR are calculated. The evaluation of sentiment-based generation relies on metrics such as accuracy, precision, recall, and the F1-score to gauge its quality. Additionally, the reward functions are extracted from *wight&bias* (Biewald, 2020).

**Table 6**

*Quality of Texts Created with SFTnRLm Models.*

Model	Perplexity ( $\downarrow$ )	SLOR ( $\uparrow$ )	FRE ( $\uparrow$ )	Coherence Score ( $\uparrow$ )
SFT1RL1	1.12	0.96	79.50	0.864
SFT1RL2	1.12	<u>0.99</u>	<u>81.48</u>	0.866
SFT2RL1	1.12	0.96	<u>75.37</u>	0.865
SFT2RL2	<u>1.11</u>	0.87	79.31	<u>0.869</u>

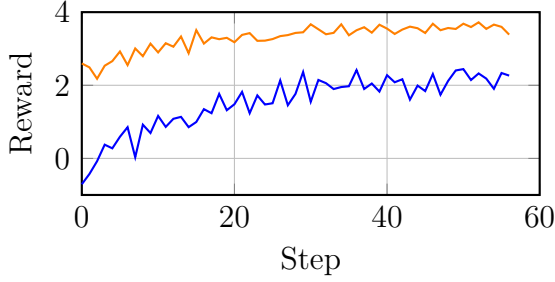
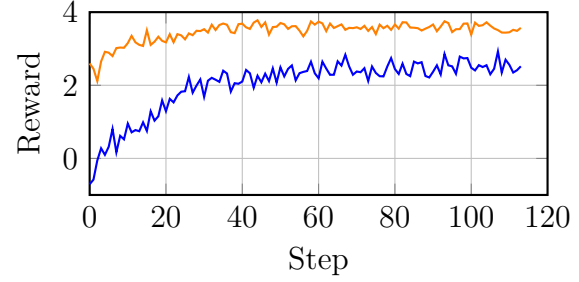
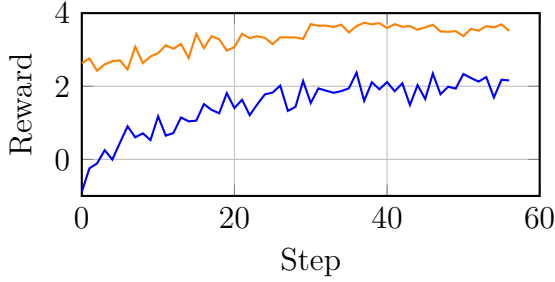
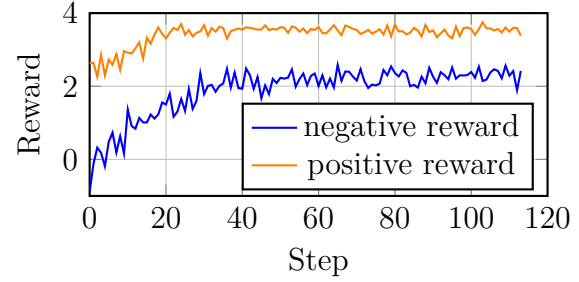
Table 6 presents the evaluation results for text quality. Perplexity, FRE, and Coherence scores are similar among the presented models and do not differ significantly from the scores obtained in the SFT process (Tab. 4). The SLOR score is similar for models optimized with RL. Additionally, it is lower than for SFT models. The influence of RL on fluency is not detectable at a significant level.

**Table 7**

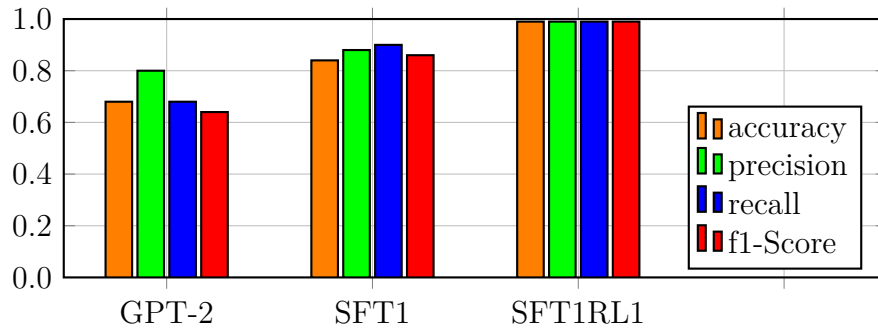
*Comparison of Sentiment Scores among GPT-2, SFT1 and SFT1RL1.*

Model	Accuracy ( $\uparrow$ )	Precision ( $\uparrow$ )	Recall ( $\uparrow$ )	F1- score ( $\uparrow$ )
SFT1RL1	0.99	0.99	0.99	0.99
SFT1RL2	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>
SFT2RL1	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>
SFT2RL2	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>

As shown in Tab. 7, the texts optimized with RL performs perfectly across all sentiment metrics. In Figure 15, reward functions are computed for two distinct tasks: 1) generating positive text (orange) and 2) generating negative text (blue). The positive reward has been consistently high since the start of training, with a slight incremental increase observed over time. It indicates that the models have already performed well on the task of generating positive text. Conversely, the models generated at the start struggled to produce negative texts proficiently. The performance improves significantly

**Figure 15***Trends of Reward Function*(a) *SFT for one epoch and RL for one epoch.*(b) *SFT for one epoch and RL for two epochs.*(c) *SFT for two epochs and RL for one epoch.*(d) *SFT for two epochs and RL for two epochs.**Note: Own Creation*

during the RL training. The average reward for this task increased from about -1 to about 2. The trends are increasing during the training phase of the first epoch (16a, 16c). At the end of this epoch, the trends are stabilized and do not improve in the second epoch. In conclusion, after the first epoch, the model learned to generate texts with a specified sentiment.

**Figure 16***Effect of Training (SFT & RL) on Sentiment Control.**Note: Own Creation*

**5.2.3 Improvement of Sentiment Control During Training.** Comparing the sentiment scores among GPT-2, SFT1 and SFT1RL1 (Fig. 16) it is notable, that SFT and RL improve the text generation in terms of sentiment control across all metrics. The first significant improvement was achieved while SFT. The second one, after the first epoch of RL.

## 5.3 Performance of Keyword Control

During the post-processing, four different decoding methods are applied with keyword control: Greedy Search, Top- $k$ , Top- $p$ , and Beam Search. Based on sample texts, the values of  $k$  and  $p$  are selected as a trade-off between Success Rate and a subjective assessment of diversity and coherence. The best results are achieved with  $k=15$  and  $p=0.5$ .

**5.3.1 Evaluation Based on Nouns.** To evaluate the overall performance of the models a list of 50 sets of keywords is generated. Every list consisted of three randomly selected nouns related to the hotel reviews since the model was fine-tuned on a hotel-related dataset. Examples of words within one set are "Lage", "Lobby" and "Fitness". The selected keywords do not inherently carry sentiment. The prompt consists of a randomly selected sentiment token (*[positive]* or *[negative]*) and sentence "Wir waren in diesem Hotel.". The same keyword set and corresponding prompts are utilized across all decoding strategies and models.

**Table 8**

*Perplexity of Texts Decoded with Different Strategies and Forcing of Nouns.*

Model	Greedy Search	Top- $k$ , $k = 15$	Top- $p$ , $p=0.5$	Beam Search
SFT1RL1	<u>1.09</u>	<u>1.09</u>	1.10	1.06
SFT1RL2	<u>1.09</u>	<u>1.09</u>	<u>1.09</u>	1.06
SFT2RL1	1.15	1.10	1.12	1.06
SFT2RL2	1.14	1.10	<u>1.09</u>	<u>1.05</u>
Avg. Fine - Tuned Models	1.12	1.09	1.10	1.06
GPT - 2	1.78	2.01	2.24	<u>1.05</u>

Perplexity is employed as the measure of fluency. As shown in Tab. 8, the Perplexity across the fine-tuned models is similar and varies between 1.05 and 1.15. The average Perplexity scores for those models demonstrate the best performance for Beam Search, as the Perplexity is the lowest. The scores for Top- $p$  and Top- $k$  strategies are lower



than for Greedy Search, indicating their better performance.

When contrasting GPT-2’s performance with that of fine-tuned models, it is evident that the GPT-2 model demonstrates higher Perplexity in Greedy Search, Top- $p$ , and Top- $k$ . This higher Perplexity suggests a lower fluency across those decoding strategies. A potential reason is that the model received an input prompt containing the sentiment token it had not been trained on.

**Table 9**  
*Influence of Decoding Strategy on Success Rate*

Model	Greedy Search	Top- $k$ , $k = 15$	Top- $p$ , $p=0.5$	Beam Search
SFT1RL1	63%	66%	<u>65%</u>	29%
SFT1RL2	59%	65%	62%	29%
SFT2RL1	50%	65%	62%	29%
SFT2RL2	55%	60%	61%	29%
Avg. Fine - Tuned Models	57%	64%	63%	29%
GPT - 2	<u>75%</u>	<u>73%</u> .	<u>65%</u>	<u>34%</u>

The performance in keyword utilization is evaluated with Success Rate. While analyzing the fine-tuned models in terms of Success Rate (Tab. 9) it is notable, that the proposed Beam Search mechanism performs poorly (29%) compared to other decoding strategies, since the Success Rate is about 30 pp. lower. Additionally, there are two obvious conclusions. First, the longer the SFTn model is trained with RL, the worse the Success Rate is for Greedy Search, Top- $k$  and Top- $p$ . The scores for Beam Search stay unchanged. In the observed scores, the Success Rates for Top- $k$  and Top- $p$  surpass that of Greedy Search. Nevertheless, an alternate trend emerges for the GPT-2. The highest average Success Rate was achieved through a Greedy Search. Another observation reveals that, on average, the Success Rate is higher for the GPT-2 model compared to the fine-tuned models. This might suggest that the created models prioritize sentiment, making the logit modification a secondary factor.

A crucial aspect of the evaluation of the approach combining sentiment and keyword control is sentiment accuracy, evaluating how well the sentiment is incorporated in text generation. As shown in Tab. 10, the keyword control with nouns do not have an impact on sentiment accuracy. The fine-tuned models still generate text with the correct sentiment. The sentiment accuracy is slightly better for Beam Search, Top- $k$  and Top- $p$  than for Greedy Search.

The sentiment for text generated by GPT-2 varies from 0.40 to 0.44, working worse than the random generation.

**Table 10***Sentiment Accuracy for Different Decoding Strategies.*

Model	Greedy Search	Top- $k$ , $k = 15$	Top- $p$ , $p=0.5$	Beam Search
SFT1RL1	0.98	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>
SFT1RL2	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>
SFT2RL1	0.96	<u>1.00</u>	0.98	<u>1.00</u>
SFT2RL2	0.96	0.98	<u>1.00</u>	<u>1.00</u>
Avg. Fine - Tuned Models	0.98	1.00	1.00	1.00
GPT - 2	0.42	0.42	0.40	0.44

**5.3.2 Evaluation Based on Sentiment-Carrying Adjectives.** Combining sentiment tokens with keywords that inherently carry their sentiment, like “*hässlich*” (ugly) and “*schön*” (beautiful), presents a challenge. These words inherently represent negative and positive sentiments, respectively, adding complexity to sentiment analysis tasks. These words might impact the sentiment score or potentially not be included in the text if they conflict with the intended sentiment.

To measure this influence, an experiment is conducted using the model SFT1RL1 with the decoding strategy Top- $k$ . This model has been selected, as the scores for both Success Rate and sentiment are highest among the proposed models and decoding strategies. There are 180 texts generated. Similarly, as for the experiment with nouns, sentiment token is given as input along with the sentence “*Wir waren in diesem Hotel.*”. 90 texts were generated with a positive sentiment token and 90 with a negative one. For every text, a keyword list is prepared that contains two adjectives and one noun. For both sentiment tokens, a group of 30 texts based on positive keywords, 30 based on negative keywords, and 30 on mixed were generated. The positive keyword set comprises two positive adjectives, such as “*schön*” (beautiful) and “*interessant*” (interesting), along with a noun e.g. “*Lobby*”. Conversely, the negative keywords encompass adjectives like “*hässlich*” (ugly), “*langweilig*” (boring), and a noun. Meanwhile, the mixed keyword category includes one positive (e.g. “*schön*”), one negative (e.g. “*hässlich*”), and a noun. The results of this experiment are shown in Tab. 11.

When comparing the overall performance of this approach to the experiment with only the utilization of nouns, it can be observed that the sentiment-based keywords worsen the performance in terms of Perplexity, sentiment accuracy, and Success Rate. On average, the Perplexity increased by 0.21 (from 1.10 to 1.31) indicating a worsened quality of fluency. The Success Rate dropped by 29 pp. (from 66% to 37%) meaning that these models are less prone to include sentiment-carrying keywords. As the sentiment

**Table 11**

*Mutual Performance of Keyword Control with Sentiment-Carrying Words and Sentiment Control.*

Sentiment Token	Positive			Negative			Overall
Keyword Group	positive	negative	mixed	positive	negative	mixed	
Perplexity ( $\downarrow$ )	1.41	1.22	<u>1.15</u>	1.56	1.24	1.19	1.31
Sentiment accuracy ( $\uparrow$ )	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	0.87	0.97	0.97	0.97
Success Rate ( $\uparrow$ )	56%	17%	32%	<u>60%</u>	26%	30%	37%

dropped only slightly (from 1.00 to 0.97), a conclusion can be drawn, that the model is more prone to preserve its sentiment than including keywords.

The common pattern exists for positive keywords for both positive and negative sentiment tokens. The Success Rate is highest for both tokens when using positive keywords, but Perplexity is the worst. It is assumed, that this may be caused by the characteristics of the training dataset. Even while writing a negative review, a user could have used positive adjectives. However, the model is able to include positive keywords but at the cost of Perplexity. Negative and mixed keywords groups have better Perplexities, but worse Success Rates. The sentiment accuracy decreases in combination with the negative sentiment token. A further decrease can be observed for positive keywords. Since the model includes those keywords, they have a negative influence on sentiment. It is difficult to include negative and mixed keywords, independently of sentiment tokens.

**5.3.3 Evaluation of Different Control Inputs.** Both models, the German GPT-2 and SFT1RL1, are tested with and without keyword control in terms of Perplexity. The comparison is prepared for models with Greedy Search, Top- $k$  and Top- $p$ . This evaluation was conducted on 50 texts, where each of them started with the sentence ‘*Wir waren in diesem Hotel.*’. Additionally, for the text generation with the SFT1RL1, the input prompt was extended by adding a sentiment token.

Based on the Tab. 12, it can be observed that adding generation constraints to the GPT-2 model in the form of keywords, sentiment (SFT1RL1), or both (SFT1RL1 with keyword control) does not significantly influence the Perplexity since it stays at a similar level. However, the Perplexity is smallest for GPT-2 without any control.

**Table 12***Comparison of Different Control Inputs.*

Model	Greedy Search	Top- $k$	Top- $p$
GPT-2	<u>1.09</u>	<u>1.09</u>	<u>1.07</u>
GPT-2 with keyword control	<u>1.09</u>	1.12	1.11
SFT1RL1	1.13	1.13	1.14
SFT1RL1 with keyword control	1.10	<u>1.09</u>	1.10

## 5.4 Analysis of Survey Results

**5.4.1 Selection of Models.** Based on the evaluation presented in the previous section, the SFT1RL1 model is selected for the text generation since this model with Top- $k$ , with  $k=15$  performs best in terms of Perplexity (1.09, Tab. 8) and Success Rate (66%, Tab. 9). It is used in following configurations:

- SFT1RL1 with Top- $k$  for the evaluation of the mutual influence of keyword and sentiment control (Row A and B, Tab, 2)
- SFT1 and SFT1RL1 for the evaluation of the influence of Fine-Tuning on sentiment control with Top- $p$  decoding. This model with Top- $p$  achieves a Perplexity of 1.10 (Tab. 8) and a Success Rate of 65% (Tab. 9), which is similar to SFT1RL1 with Top- $k$ . This decoding strategy is selected due to the diversity of selected texts. It was observed that this model with Top- $k$  generates always the same text given a sentiment token and an input prompt (Row C and D, Tab, 2).
- SFT1RL1 for the evaluation of the influence of decoding strategy and keyword control (Row E and F, Tab. 2) .

**5.4.2 Demographics.** The survey was sent via social media to German-speaking persons and was evaluated by 47 individuals. As shown in Fig. 17a, the majority of the users are between 30 and 40 (19 participants) and between 40 and 60 (18 participants).

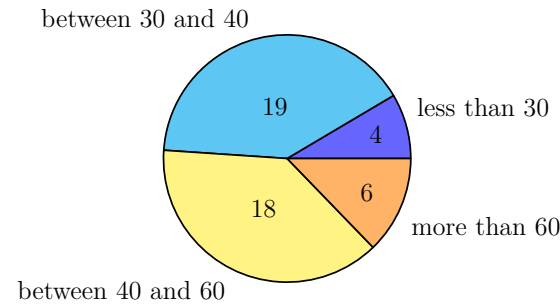
The smallest group was users under 30 - four participants. Six participants are older than 60.

The majority of the participants are men - 35 participants. 11 participants are female and one diverse, as shown in Fig. 17b.

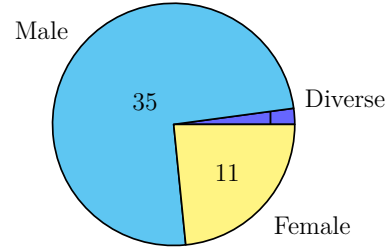
17 participants evaluated the *yellow* path, 16 the *green* and 14 the *violet* ones. The number of participants among groups is not equal. So, 14 randomly selected submissions are selected for the evaluation of every path.

**Figure 17**  
*Demographics*

(a) *Age structure of the Survey Responders*



(b) *Gender structure of the Survey Responders*



*Note: Own Creation*

**5.4.3 Inter-Annotator Agreement.** To assess the alignment of rating scores among participants in the survey, the Multi- $\kappa$  coefficient was computed for each survey. The score for the yellow path amounts to 0.18, for green to 0.14, and for violet to 0.18. As the coefficients are pretty low, they suggest a slight agreement between the participants.

**5.4.4 Evaluation of Cronbach Alpha.** During the evaluation of Cronbach's Alpha, the  $\alpha$  for fluency items amounted to 0.75 and  $\alpha$  for coherence 0.65. It indicates respectively good and moderate reliability of the groups. In the scope of the thesis, the scores are accepted as the commonly desired threshold equals 0.7. The latent variables are created based on the items. Means and standard deviation for every statement are presented in Appendix H.

**5.4.5 Evaluation of Mutual Influence of Keyword and Sentiment Control.** It is challenging to interpret the outcomes of sentiment agreement for texts generated

**Table 13**

*Sentiment Perception of Texts Generated with Sentiment Token and Sentiment - Carrying Keywords.*

	Descriptives (mean, standard deviation)			Average
	Keywords' Sentiment			
	negative	positive	mixed	
Positive Sentiment Token	3.64 (1.15)	4.43 (0.94)	3.57 (1.09)	3.88 (1.06)
Negative Sentiment Token	5.00 (0.00)	5.00 (0.00)	5.00 (0.00)	5.00 (0.00)

with negative sentiment tokens (Tab. 13). While examining the statement “The text is positive” and achieving flawless scores for each keyword group during analysis, an unforeseen contradiction emerges upon closer inspection. It poses a dual interpretation. One possibility is that the text is entirely negative, leaving no ambiguity in the readers’ perception of the sentiment. Alternatively, participants might have misinterpreted the question, responding in a manner unintended by the inquiry, consequently refuting the hypothesis regarding the positivity of the text. Consequently, the outcomes for negative sentiment token should be treated with caution.

The mean score for positive keywords and a positive token corresponds to “strongly agree” (4.43) and indicates that the sentiment alignment is performing well. However, introducing negative-based (3.64) or mixed (3.57) adjectives as keywords seems to reduce the accuracy of sentiment identification, as the participants on average “agree”.

**Table 14**

*Influence of Sentiment Token and Sentiment - Carrying Keywords on Fluency Perception.*

	Descriptives (mean, standard deviation) Keywords’ Sentiment			Average
	negative	positive	mixed	
Positive Sentiment Token	2.41(0.51)	2.54 (0.86)	2.36 (0.75)	2.44 (0.72)
Negative Sentiment Token	2.43 (0.62)	1.91 (0.51)	1.78 (0.59)	2.04 (0.57)

Table 14 presents the evaluation of fluency. All texts generated with positive sentiment tokens achieve similar, rather poor (2.44), fluency scores independently from the sentiment of the keyword. The fluency of texts generated with negative sentiment tokens is even lower (2.04).

**Table 15**

*Human Evaluation of Coherence of Texts Generated with Sentiment Token and Sentiment - Carrying Keywords.*

	Descriptives (mean, standard deviation) Model			Average
	negative	positive	mixed	
Positive Sentiment Token	2.46 (0.85)	2.86 (0.87)	2.29 (0.66)	2.52 (0.79)
Negative Sentiment Token	2.11 (0.76)	2.14 (0.72)	1.79 (0.70)	2.01 (0.73)

The mean coherence scores are presented in Tab. 15. According to participants’

ratings, the most coherent texts with a specified sentiment token are those where the text sentiment token aligns with the keyword sentiment (2.86, 2.11), however, the users disagree that they are coherent. Adding keywords with mixed sentiment yields a lower coherence perception.

**5.4.6 Influence of Fine-Tuning on Sentiment Control.** Table 16 presents the evaluation of sentiment for the GPT-2, SFT1, and SFT1RL1 models. The users on average agree and strongly agree that the sentiment of the texts matches the given one. It indicates, that SFT1 and SFT1RL1 can distinguish between sentiments and thus generate sentiment-based texts.

**Table 16**

*Perception of Sentiment of Texts Generated by Models Controlled with Sentiment Token.*

	Descriptives (mean, standard deviation)			Average
	GPT-2	Model SFT1	SFT1RL1	
Positive Sentiment Token	4.21 (1.43)	3.86 (0.95)	4.79 (0.43)	4.29 (0.94)
Negative Sentiment Token	1.36 (0.63)	4.93 (0.27)	5.00 (0.00)	3.76 (0.30)

An intriguing observation pertains to the sentiment scores obtained from GPT-2 across both positive and negative tokens, revealing a consistently high positive sentiment (the users disagree that the sentiment matches the specified one). This outcome can be attributed to two main factors. Firstly, GPT-2 lacks fine-tuning specifically for sentiment tokens, affecting its ability to discern and express sentiment accurately. Secondly, the model might tend to generate positive texts.

**Table 17**

*Fluency’s Perception of Texts Generated by Models Controlled with Sentiment Token.*

	Descriptives (mean, standard deviation)			Average
	GPT-2	Model SFT1	SFT1RL1	
Positive Sentiment Token	4.14 (0.82)	3.46 (0.88)	3.50 (0.83)	3.70 (0.84)
Negative Sentiment Token	2.54 (0.71)	2.39 (0.79)	3.79 (0.66)	2.91 (0.72)

The evaluation of fluency is presented in Tab. 17. The text generated with SFT1RL1 received a good score for both sentiments. The participants on average agree that the

text is fluent. The fluency of the text generated with the positive sentiment token (3.70) is higher than the text generated with a negative one (2.91). The users’ reviews indicate that the fluency of the texts generated by SFT1 and SFT1RL1 with the positive token is lower than GPT-2 generated text with the same token. On the other side, the text generated by the SFT1RL1 model with a negative sentiment token is perceived as more fluent than GPT-2. It is worth mentioning, that the difference between fluency for both tokens is the smallest for the SFT1RL1 model. It indicates, that the model generates text with consistent fluency independently from the choice of the sentiment token.

**Table 18**

*Perception of Influence of Fine-Tuning on Coherence.*

	Descriptives (mean, standard deviation)			Average
	GPT-2	Model SFT1	SFT1RL1	
Positive Sentiment Token	3.36 (1.33)	3.25 (0.75)	3.21 (0.77)	3.27 (0.95)
Negative Sentiment Token	2.43 (0.73)	2.11 (0.81)	3.25 (1.18)	2.60 (1.42)

Analyzing the Tab. 18, it can be observed, that texts produced with positive sentiment tokens demonstrate higher coherence levels than text generated with negative sentiment tokens by GPT-2 and SFT1. The users are on average neutral about the coherence of the positive texts, whereas when evaluating negative texts, they disagree that they are coherent. The users perceive both texts generated by SFT1RL1 as similarly coherent. They are neutral about their coherence.

**5.4.7 Influence of Decoding Strategy and Keyword Control.** The last part of the survey describes the influence of the decoding strategy on text generation. Based on Tab. 19, the average sentiment score for the text generated with the positive sentiment token is high for all decoding strategies (4.48). The highest sentiment score is obtained by the Top- $k$  decoding strategy. The participants strongly agree that all generated texts match the specified sentiment as the sentiment agreement scores are high.

Evaluation of the mean fluency score (Tab. 20) obtained for the survey reveals, that the texts of the worst perceived fluency are generated by the model with Greedy Search decoding (2.66). Overall, the best performance in terms of fluency shows the model with Top- $p$ . However, the best score for the positive sentiment token is obtained with Top- $k$  (3.27), and for the negative sentiment token, it is obtained by Top- $p$  (3.73).

Greedy Search performs poorly for both sentiments (Tab. 21). The Top- $k$  strategy performed better for positive generated text than for negative one, Top- $p$  worked exactly



**Table 19***Sentiments Perception of Texts Decoded with Different Strategies.*

	Descriptives (mean, standard deviation) Decoding Strategy			Average
	Greedy Search	Top- $k$ , $k=15$	Top- $p$ , $p=0.5$	
Positive Sentiment Token	4.29 (1.14)	4.79 (0.58)	4.36 (0.63)	4.48 (0.78)
Negative Sentiment Token	5.00 (0.00)	4.71 (0.47)	4.79 (0.80)	4.83 (0.42)

**Table 20***Influence of Decoding Strategies on the Perception of Generated Texts.*

	Descriptives (mean, standard deviation) Decoding Strategy			Average
	Greedy Search	Top- $k$ , $k=15$	Top- $p$ , $p=0.5$	
Positive Sentiment Token	2.66 (0.58)	3.27 (1.02)	3.00 (0.63)	2.98 (0.74)
Negative Sentiment Token	2.66 (0.82)	2.66 (0.75)	3.73 (0.93)	3.02 (0.83)

**Table 21***Influence on Coherence's Perception of Texts Decoded with Different Strategies.*

	Descriptives (mean, standard deviation) Decoding Strategy			Average
	Greedy Search	Top- $k$ , $k=15$	Top- $p$ , $p=0.5$	
Positive Sentiment Token	2.39 (0.83)	3.11 (1.00)	2.96 (0.77)	2.82 (0.87)
Negative Sentiment Token	2.50 (0.98)	2.43 (0.75)	3.21 (0.80)	2.71 (0.84)

in the opposite way. Consequently, employing a decoding strategy that considers a larger number of potential new words during generation results in text that appears more coherent to human evaluators.

## 6 Discussion

This chapter presents answers to the predefined research questions. Section 6.1 deals with the possibility of a combination of keyword and sentiment control into one model.

Section 6.2 showcases the user’s perception of the proposed controlling approaches compared to GPT-2. Section 6.3 contains the comparison of Perplexity for GPT-2 and different proposed controlling approaches. Lastly, Sect. 6.4 reveals how the Perplexity and Success Rate differ for different decoding strategies.

## 6.1 Research Question 1

*Can sentiment and keyword control be effectively combined into one model?*

Combining sentiment and keyword control within a single text generation model showcases feasibility but presents substantial challenges. While the model effectively aligns sentiment with non-sentimental and sentimental keywords, it compromises text quality, affecting fluency, coherence, and sentiment accuracy (see Tab. 8 - 11). GPT-2, although not trained for this task, demonstrates greater ease in incorporating diverse keywords, highlighting the complexities involved in effectively balancing these dual influences within a cohesive model. This feasibility underscores the intricate nature of merging sentiment and keywords, revealing the need for nuanced approaches to achieve optimal text generation quality.

Upon analyzing the mutual influence between keywords (nouns) and sentiment control utilizing the SFT1RL1 model with the top- $k$  decoding strategy ( $k=15$ ), several observations become apparent. The model can include both keywords (nouns) and sentiment tokens as the Success Rate exceeds 0.5 (66%, Tab. 9), suggesting that the inclusion of keywords is not arbitrary, potentially implying deliberate inclusion. Utilization of sentiment-carrying keywords leads to a decrease in Success Rate (of 37%, Tab. 11).

Another important insight comes from human evaluation. The model using nouns as keywords achieves a good score (4.79, Tab. 19) in terms of sentiment agreement for the positive token. Also, the rating for a text generated with a negative sentiment token is perceived as matching the sentiment. The model was rated with 3.27 and 2.66 for positive and negative tokens respectively (Tab. 20) in terms of fluency indicating a neutral perception of the fluency. The coherence score was 3.11 for the positive sentiment token and 2.43 for the negative one (Tab. 18) indicating moderately satisfying text coherence as the average ratings are neutral.

The texts generated with sentiment and sentiment-carrying keywords were perceived as more unsatisfying in terms of sentiment, fluency, and coherence as their means are lower.

## 6.2 Research Question 2

*How do users perceive the text, which is generated by the proposed model, compared to a text generated by GPT with respect to a chosen sentiment?*

The GPT model acts as a benchmark for assessing the proposed model, as it is employed in the framework of this research to develop sentiment-based models. Due to this, the texts generated by GPT-2, SFT1RL1 with only sentiment control, and SFT1RL1 with additional keyword control (Top- $p$  decoding,  $p=0.5$ ) are compared. Texts are generated with both positive and negative sentiment tokens. The visual summary of the relevant data from the survey is presented in Fig. 18.

Starting with texts generated with positive sentiment tokens, GPT-2 achieved a sentiment agreement mean of 4.21 (Tab. 16) meaning, that the participants strongly agree that it matches the sentiment. It was also perceived on average as fluent (4.14, Tab. 17). The coherence of the text achieved 3.36 (Tab. 18) meaning rather neutral perception. The SFT1RL1 model achieved a mean sentiment agreement score of 4.79 (Tab. 16), a fluency score of 3.50 (Tab. 17), and a coherence score of 3.21 (Tab. 18). Text generated by the SFT1RL1 control is perceived as matching the sentiment better, than GPT-2, however, the fluency and coherence are decreased.

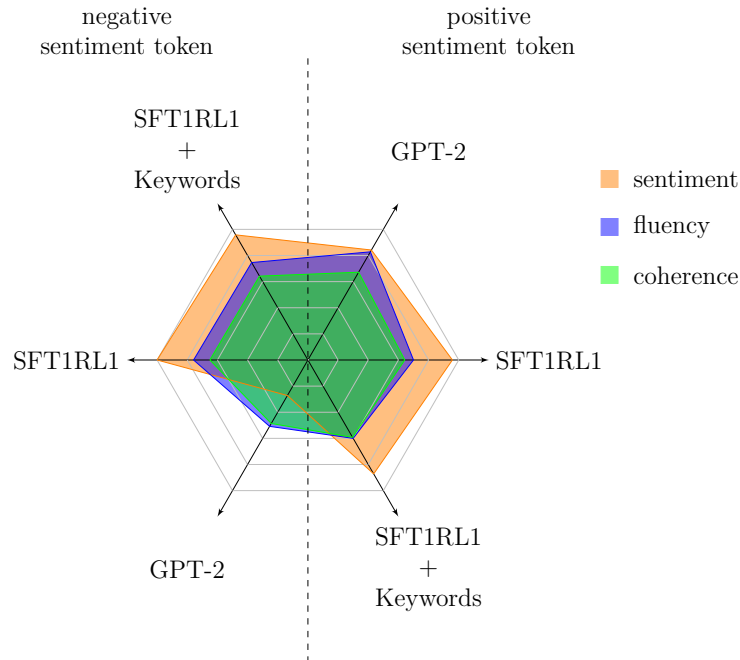
After adding the keyword control to this model, the mean sentiment score amounts to 4.36 (Tab. 19), fluency to 3.00 (Tab. 20), and coherence to 2.96 (Tab. 21). The text generated by the SFT1RL1 model with keyword control is perceived as matching closer sentiment than GPT-2, indicating better sentiment-generation performance. However, adding more control over keywords worsens the fluency and coherence perception and thus the text quality.

The users assign a sentiment score of 1.36 (Tab. 16) to the text produced using the GPT-2 model with a focus on sentiment token. This indicates, that the text in their opinion does not match the sentiment. The text is rather not fluent (2.54, Tab. 17), and not coherent (2.43, Tab. 18). The text generated by the SFT1RL1 model obtained a sentiment score of 5.00 (Tab. 16) and thus was perceived as matching. However, due to the ambiguity of the statement, the extent of the agreement cannot be interpreted. The fluency was awarded with a score of 3.79 (Tab. 17) and the coherence with 3.25 (Tab. 18). The text generated by this model matches the chosen sentiment better, than the text generated by GPT-2. Additionally, it is more fluent and less coherent than the text generated by GPT-2.

Text generated with the SFT1RL1 model with keyword control obtained a sentiment

**Figure 18**

*Perception of Text Generated with Sentiment and Keyword Control and GPT-2.*



*Note: Own Creation.*

score of 4.79 (Tab. 19); the users agree that the sentiment is generated correctly. It was neutral fluent (3.73, Tab. 20) and coherent (3.21, Tab. 21). The text with forced keywords matches the sentiment but achieves worse fluency and better coherence than the text without forced keywords.

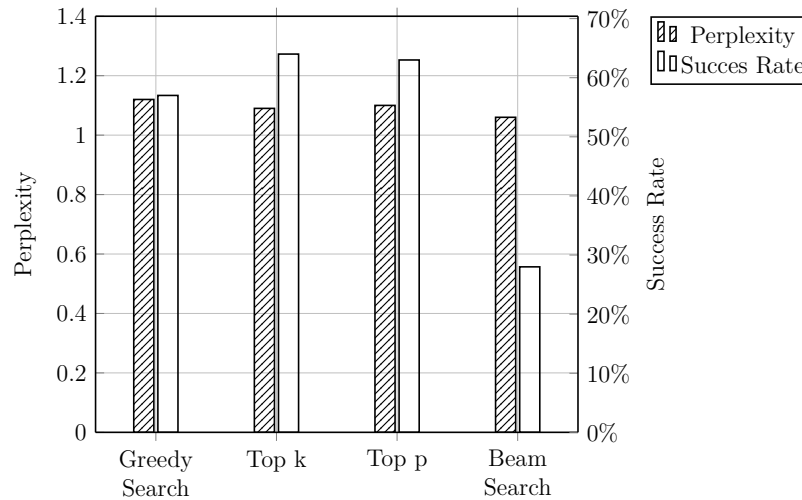
The user's rating can indicate, that texts generated by the sentiment-based model both with and without keyword control match better the chosen sentiment than the GPT-2 model. In the author's opinion, no general conclusion can be taken for fluency and coherence as no common pattern in the evaluation can be found.

### 6.3 Research Question 3

*How is the Perplexity for models with control generation compared to the original GPT?*

Referring to Table 12, the incorporation of generation constraints, through keywords, sentiment (SFT1RL1), or a combination of both (SFT1RL1 with keyword control)—into the GPT-2 model does not notably impact Perplexity. It remains consistently at a similar level across these variations. Notably, the smallest Perplexity is observed in the case of GPT-2 without any form of control.

**Figure 19**  
*Summarized Comparision of Decoding Strategies.*



*Note: Own Creation.*

## 6.4 Research Question 4

*What are the differences in Perplexity and keyword utilization using different decoding approaches?*

The analysis of Perplexity across various decoding approaches within sentiment-based models demonstrates a striking similarity among the four strategies: Greedy Search, Top- $k$ , Top- $p$ , and Beam Search (summarized in Fig. 19). The average Perplexity values over the fine-tuned sentiment-based model reveal minimal differences.

This similarity in Perplexity values across decoding strategies suggests that, in terms of language prediction, all four methods perform comparably within the context of sentiment-based models. Despite slight variations, the overall Perplexity remains consistent, indicating similar predictive capabilities across these decoding strategies when applied to fine-tuned sentiment-based models.

In the analysis of average Success Rates among sentiment-controlled models, a consistent trend emerges: Top- $k$  sampling achieved the highest average Success Rate at 64%, closely followed by Top- $p$  at 63%, and then Greedy Search at 57%. Notably, the proposed Beam Search performed poorly, obtaining a low Success Rate. These findings indicate that sampling methods like Top- $k$  and Top- $p$  tend to outperform greedy approaches such as Greedy Search. Therefore, the adoption of token sampling strategies appears pivotal in achieving higher Success Rates in sentiment-controlled text generation models.

## 7 Conclusion and Outlook

### 7.1 Conclusion

The objective of this thesis was to prepare a pipeline that allowed control over text with sentiment and keywords. It is possible to control both with the proposed pipeline. So, the thesis objective is fulfilled. Nevertheless, the methods do not support each other and the mutual control reduces the readers' satisfaction, as they rated the fluency and coherence low. The survey's participants believe that the human interaction in the texts could improve the overall quality (Fig. 22, Appendix I). Thus, the proposed pipeline can serve as a tool that helps in text generation in creative areas such as marketing. However, the generated texts require a user to correct the grammar and spelling.

In the course of the thesis, a couple of methods are utilized: Sentiment Discriminator, SFT, RL, and decoding strategies for keyword generation. Every step is followed by evaluation.

In the first step, different sentiment discriminators are created and evaluated. Models were trained on datasets cleaned in different ways. Two architectures were prepared using BERT layers as the feature extractor. It can be concluded, that the extent of cleaning influences the model performance. The models achieved the best performance by being trained on the data set that was not extensively cleaned. In the author's opinion this may be due to the implemented self-attention mechanism; during the extensive cleaning important contextual information may be lost and the BERT model is not able to find patterns.

The SFT allows Fine-Tuning the GPT-2 model on the task of sentiment-controlled text generation. The RL optimization improves the sentiment accuracy of generated text. After training for one epoch with SFT, an improvement in the sentiment accuracy by 16 pp. is achieved compared to GPT-2. However, more training does not yield significant improvement. For further improvement, the RL utilizing PPO is employed. The sentiment of those texts is correct, as proven by the sentiment accuracy of 1. This is a significant achievement. It outperforms the results achieved by Dathathri et al. (2019) in their research. However, it is also difficult to compare, as the texts were generated in different languages (German and English) and evaluated with different trained models. It could be interesting to compare the Perplexity of proposed models, however, it is impossible due to different generation languages, resulting in different models used for evaluation.

After the sentiment control is successfully set up, the keyword control mechanism

is implemented. The approach proposed by Pascual et al. (2020) is extended to other decoding strategies and to the control over keywords independently from their order. (Pascual et al., 2020) achieved promising performance in terms of Success Rate, so it was expected, that the extended approach would also perform satisfactorily. Unfortunately, it achieves worse performance for all decoding strategies. Also, the generation time is long and not satisfying. The corresponding Success Rate is worse compared to GPT-2. The author assumes that the models that are heavily trained for text sentiment tend to prioritize sentiment utilization higher than forcing specific keywords. Adding sentiment-carrying keywords may affect the sentiment accuracy. Since the presented models are fine-tuned on the dataset strongly focusing on reviews, the models may tend to prioritize words and phrases being more popular in the test dataset. This may make the keyword control mechanism secondary when it is implemented as a part of the post-processing mechanism.

The analysis of human perception of the generated texts gives new insights into fluency and coherence. The texts generated by the sentiment-based trained model were perceived as moderately fluent and moderately coherent. Adding sentiment-carrying keywords results in lower fluency and lower coherence. However, adding keywords does not significantly influence the perception of sentiment; the texts still match to specifications. The Perplexity and coherence scores are satisfactory. However, the survey results show a different performance. This highlights the limitation of relying solely on automated metrics. Human evaluation of text remains crucial. The majority of participants confirm that human post-processing after the automated text generation will improve its quality (Appendix I). This finding supports the author’s opinion that the model may support creative content generation but as of the time of this thesis, the model is not able to fulfill the task on its own.

The author sees also a benefit in the repetition of this survey which would include a higher number of texts so that the results will be stable. Additionally, a greater number of participants would be beneficial, as the author doubts that the number of participants in the conducted survey led to significant results. However, a more complex survey was not possible at the time of this research due to time and resource constraints. For the evaluation of coherence, more items should be added, since Cronbach’s Alpha test resulted in a score  $\alpha$  equaled to 0.65 indicating moderate reliability.

## 7.2 Outlook

The control over used keywords still requires further research and improvements. It is evident that distributing controls across various sections of the model’s architecture,

including fine-tuning and post-processing, does not meet all the specified requirements. Since the sentiment control approach showed promising results, future research steps could involve combining the two controls into one model in a similar way to the sentiment control. The author sees three possible directions for the utilization of RL approach. The first employs a combination of sentiment discriminators for rewarding/penalizing sentiment generation and calculation of Success Rate for optimization of keywords into one reward function. However, the design of a suitable reward function that balances sentiment and keywords may be a challenging task. The second analysis shows that the users may evaluate texts differently than the automated metrics. Thus, it may be beneficial to optimize a model with human feedback, where the user’s rating would serve directly as a reward. The third possibility would be the combination of those two approaches, which benefits from both.

Also Fine-Tuning in a Few-Shot fashion could be of advantage. By leveraging the robust representations learned during pre-training and adapting the model to our specific control requirements using a limited number of labeled instances, Few-Shot Fine-Tuning offers a targeted approach. This method aims to enhance the model’s capability to incorporate both sentiment and keyword control seamlessly into the text generation process.

Both RL optimization and Few-Shot Fine-Tuning hold promise in addressing the challenges encountered in prior attempts, where methods lacked mutual support or adversely affected text fluency and coherence. This approach aims to empower the model to learn and adapt quickly to the simultaneous control objectives of sentiment and keywords, potentially mitigating the drawbacks observed in earlier iterations. Since the controls would be learned during the training, the inference process could be faster. This would be beneficial in real-life applications and improve the user experience.

Further, another questionnaire may include human-written text samples that identify the user perception. With that baseline, the analysis may not be restricted to the assumption that values (of e.g. coherence or fluency) of good texts are higher than 3.40 as it is in this thesis.

Additionally, the dataset could be extended to more different topics, since the presented approach was trained basically on hotels and movie reviews. Adding more diverse data could also encourage the model to create a more creative text. During this research, the language of the model is German. The variety of models in this language is smaller than in English. Additionally, the utilization of SOTA models as GPT-4.5 could be beneficial.



## References

- Amiri, R., Mehrpouyan, H., Fridman, L., Mallik, R., Nallanathan, A., & Matolak, D. (2018). A machine learning approach for power allocation in hetnets considering qos. <https://doi.org/10.1109/ICC.2018.8422864>
- Artstein, R. (2017). Inter-annotator agreement. *Handbook of linguistic annotation* (pp. 297–313). Springer.
- Biewald, L. (2020). Experiment tracking with weights and biases [Software available from wandb.com]. <https://www.wandb.com/>
- Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with python: Analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”.
- Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan), 993–1022.
- Carifio, J., & Perla, R. J. (2007). Ten common misunderstandings, misconceptions, persistent myths and urban legends about likert scales and likert response formats and their antidotes. *Journal of social sciences*, 3(3), 106–116. <https://www.researchgate.net/publication/26619168>
- Chaumond, J. (2020). + dbmdz german bert models. <https://huggingface.co/dbmdz/bert-base-german-cased>
- Collobert, R., Kavukcuoglu, K., & Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. *BigLearn, NIPS Workshop*.
- Dang, N. C., Moreno-García, M. N., & De la Prieta, F. (2020). Sentiment analysis based on deep learning: A comparative study. *Electronics*, 9(3), 483. <https://doi.org/10.3390/electronics9030483>
- Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., & Liu, R. (2019). Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*. <https://doi.org/10.48550/arXiv.1912.02164>
- Davies, M., & Fleiss, J. L. (1982). Measuring agreement for multinomial data. *Biometrics*, 38(4), 1047–1051.
- De Silva, M. (2023). Preprocessing steps for natural language processing (nlp): A beginner’s guide [Accessed on December 14, 2023]. <https://medium.com/@maleeshadesilva21/preprocessing-steps-for-natural-language-processing-nlp-a-beginners-guide-d6d9bf7689c9>

- Deep, A. (2020). Word2vec, glove, fasttext and baseline word embeddings step by step [Accessed on November 9, 2023]. <https://medium.com/analytics-vidhya/word2vec-glove-fasttext-and-baseline-word-embeddings-step-by-step-d0489c15d10b>
- Delovski, B. (2023). Intro to transformers: The decoder block [Accessed on Oct 26, 2023]. [https://www.edlitera.com/blog/posts/transformers-decoder-block#mcetoc\\_1gtef7bgt11](https://www.edlitera.com/blog/posts/transformers-decoder-block#mcetoc_1gtef7bgt11)
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. <https://doi.org/10.48550/arXiv.1810.04805>
- Fan, A., Lavril, T., Grave, E., Joulin, A., & Sukhbaatar, S. (2020). Addressing some limitations of transformers with feedback memory. *arXiv preprint arXiv:2002.09402*. <https://doi.org/10.48550/arXiv.2002.09402>
- Fan, A., Lewis, M., & Dauphin, Y. (2018). Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*. <https://doi.org/10.48550/arXiv.1805.04833>
- FastText. (n.d.). Fasttext [Accessed on May 15, 2023]. <https://fasttext.cc/>
- Flesch, R. (1948). A new readability yardstick. *Journal of applied psychology*, 32(3), 221.
- Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*. <https://doi.org/10.48550/arXiv.1308.0850>
- Gugger, S. (2020). Distilbert-base-german-cased. <https://huggingface.co/distilbert-base-german-cased>
- Guhr, O., Schumann, A.-K., Bahrmann, F., & Böhme, H. J. (2020a). Broad-coverage german sentiment classification model for dialog systems [Accessed on May 15, 2023]. <https://github.com/oliverguhr/german-sentiment>
- Guhr, O., Schumann, A.-K., Bahrmann, F., & Böhme, H. J. (2020b). Training a broad-coverage German sentiment classification model for dialog systems. In N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odiijk, & S. Piperidis (Eds.), *Proceedings of the twelfth language resources and evaluation conference* (pp. 1627–1632). European Language Resources Association. <https://aclanthology.org/2020.lrec-1.202>
- He, X. (2021). Parallel refinements for lexically constrained text generation with bart. *arXiv preprint arXiv:2109.12487*. <https://doi.org/10.48550/arXiv.2109.12487>
- Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2019). The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*. <https://doi.org/10.48550/arXiv.1904.09751>

- Honnibal, M., & Montani, I. (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing* [To appear].
- Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., & Xing, E. P. (2017). Toward controlled generation of text. *International conference on machine learning*, 1587–1596. <https://doi.org/10.48550/arXiv.1703.00955>
- Hugging Face. (n.d.-a). <https://huggingface.co/>
- Hugging Face. (n.d.-b). Perplexity of fixed-length models. <https://huggingface.co/docs/transformers/perplexity>
- Hugging Face. (n.d.-c). Text classification [Accessed on December 15, 2023]. [https://huggingface.co/docs/transformers/tasks/sequence\\_classification](https://huggingface.co/docs/transformers/tasks/sequence_classification)
- Kann, K., Rothe, S., & Filippova, K. (2018). Sentence-level fluency evaluation: References help, but can be spared! *arXiv preprint arXiv:1809.08731*. <https://doi.org/10.48550/arXiv.1809.08731>
- Kapadia, S. (2019). Evaluate topic models: Latent dirichlet allocation (lda) [Accessed on November 26, 2023]. <https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0>
- Krause, B., Gotmare, A. D., McCann, B., Keskar, N. S., Joty, S., Socher, R., & Rajani, N. F. (2020). Gedi: Generative discriminator guided sequence generation. *arXiv preprint arXiv:2009.06367*. <https://doi.org/10.48550/arXiv.2009.06367>
- Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17), 1–5. <http://jmlr.org/papers/v18/16-365.html>
- Lhoest, Q., Villanova del Moral, A., Jernite, Y., Thakur, A., von Platen, P., Patil, S., Chaumond, J., Drame, M., Plu, J., Tunstall, L., Davison, J., Šaško, M., Chhablani, G., Malik, B., Brandeis, S., Le Scao, T., Sanh, V., Xu, C., Patry, N., ... Wolf, T. (2021). Datasets: A community library for natural language processing. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 175–184. <https://aclanthology.org/2021.emnlp-demo.21>
- Li, J., Tang, T., Zhao, W. X., Nie, J.-Y., & Wen, J.-R. (2022). Pretrained language models for text generation: A survey. *arXiv preprint arXiv:2201.05273*. <https://doi.org/10.48550/arXiv.2105.10311>
- Lipenkova, J. (2022). Choosing the right language model for your nlp use case [Accessed on May 15, 2023]. <https://towardsdatascience.com/choosing-the-right-language-model-for-your-nlp-use-case-1288ef3c4929>

- Lokare, G. (2023). Preparing text data for transformers: Tokenization, mapping and padding [Accessed on November 7, 2023]. <https://medium.com/@lokaregns/preparing-text-data-for-transformers-tokenization-mapping-and-padding-9fbfbce28028>
- López Espejel, J. (2022). Understanding greedy search and beam search [Accessed on December 3, 2023]. [https://medium.com/@jessica\\_lopez/understanding-greedy-search-and-beam-search-98c1e3cd821d](https://medium.com/@jessica_lopez/understanding-greedy-search-and-beam-search-98c1e3cd821d)
- Manocchio, L. D., Layeghy, S., Lo, W. W., Kulatilleke, G. K., Sarhan, M., & Portmann, M. (2023). Flowtransformer: A transformer framework for flow-based network intrusion detection systems. *arXiv preprint arXiv:2304.14746*. <https://doi.org/10.48550/arXiv.2304.14746>
- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, ... Xiaoqiang Zheng. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems [Software available from tensorflow.org]. <https://www.tensorflow.org/>
- Martinez, J. J. (2023). Supervised fine-tuning: Customizing llms. <https://medium.com/mantisnlp/supervised-fine-tuning-customizing-llms-a2c1edbf22c3>
- McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). <https://doi.org/10.25080/Majora-92bf1922-00a>
- Minixhofer, B., Paischer, F., & Rekabsaz, N. (2022). WECHSEL: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models. *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 3992–4006. <https://aclanthology.org/2022.naacl-main.293>
- pandas development team, T. (2020). *Pandas-dev/pandas: Pandas* (Version latest). Zenodo. <https://doi.org/10.5281/zenodo.3509134>
- Pascual, D., Egressy, B., Bolli, F., & Wattenhofer, R. (2020). Directed beam search: Plug-and-play lexically constrained language generation. *arXiv preprint arXiv:2012.15416*. <https://doi.org/10.48550/arXiv.2012.15416>
- Pascual, D., Egressy, B., Meister, C., Cotterell, R., & Wattenhofer, R. (2021). A plug-and-play method for controlled text generation. *arXiv preprint arXiv:2109.09707*. <https://doi.org/10.48550/arXiv.2109.09707>

- Pecánek, M. (2021). Flesch reading ease: Does it matter for seo? (data study). <https://ahrefs.com/blog/flesch-reading-ease/>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct), 2825–2830. <https://doi.org/10.48550/arXiv.1201.0490>
- Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. <https://doi.org/10.3115/v1/D14-1162>
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding with unsupervised learning.
- Rehurek, R., & Sojka, P. (2011). Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2).
- Reiter, E., & Dale, R. (2002). Building applied natural language generation systems. *Natural Language Engineering*, 3. <https://doi.org/10.1017/S1351324997001502>
- Röder, M., Both, A., & Hinneburg, A. (2015). Exploring the space of topic coherence measures. *Proceedings of the eighth ACM international conference on Web search and data mining*, 399–408. <https://doi.org/10.1145/2684822.2685324>
- Samuels, A., & Mcgonical, J. (2020). News sentiment analysis. *arXiv preprint arXiv:2007.02238*. <https://arxiv.org/pdf/2007.02238.pdf>
- Schmid, H. (1999). Improvements in part-of-speech tagging with an application to german. *Natural language processing using very large corpora* (pp. 13–25). Springer.
- Schmid, H. (2013). Probabilistic part-of-speech tagging using decision trees. *New methods in language processing*, 154.
- Schmid, H. (n.d.). Treetagger - a part-of-speech tagger for many languages [Accessed on November 9, 2023]. <https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/>
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Schweter, S. (2020). *German gpt-2 model* (Version 1.0.0). Zenodo. <https://doi.org/10.5281/zenodo.4275046>

- Stewart, M. (2023). Guide to classification on imbalanced datasets [Accessed on November 7, 2023]. <https://towardsdatascience.com/guide-to-classification-on-imbalanced-datasets-d6653aa5fa23>
- Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to fine-tune bert for text classification? *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, 194–206. <https://doi.org/10.48550/arXiv.1905.05583>
- SuperAnnotate. (2023). Fine-tuning large language models (llms) in 2023. <https://www.superannotate.com/blog/llm-fine-tuning>
- Sutton, R. S., & Barto, A. G. (n.d.). *Reinforcement learning: An introduction*. Bradford Book.
- Textstat. (n.d.). <https://pypi.org/project/textstat/>
- Tunstall, L., von Werra, L., & Wolf, T. (2022). *Natural language processing with transformers. building language applications with hugging face*. O'Reilly Media, Inc.
- Vajjala, S., Majumder, B., Gupta, A., & Surana, H. (2020). *Practical natural language processing. a comprehensive guide to building real-world nlp systems*. O'Reilly Media, Inc.
- Vallat, R. (2018). Pingouin: Statistics in python. *The Journal of Open Source Software*, 3(31), 1026. <https://doi.org/10.21105/joss.01026>
- van der Lee, C., Gatt, A., van Miltenburg, E., & Krahmer, E. (2021). Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech & Language*, 67, 101151. <https://doi.org/10.1016/j.csl.2020.101151>
- van Heeswijk, W. (2022). Proximal policy optimization (ppo) explained [Accessed on November 7, 2023]. <https://towardsdatascience.com/proximal-policy-optimization-ppo-explained-abed1952457b>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. <https://doi.org/10.48550/arXiv.1706.03762>
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental

- Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
- von Platen, P. (2020). How to generate text: Using different decoding methods for language generation with transformers [Accessed on November 5, 2023]. <https://huggingface.co/blog/how-to-generate>
- von Werra, L. (2023). Tune gpt2 to generate controlled sentiment reviews. <https://github.com/huggingface/trl/blob/main/examples/notebooks/gpt2-sentiment-control.ipynb>
- von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., & Huang, S. (2020). Trl: Transformer reinforcement learning. <https://github.com/huggingface/trl>
- Vujovic, Ž. Đ. (2021). Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications*, 12(6). <https://doi.org/10.14569/IJACSA.2021.0120670>
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., ... Rush, A. (2020). Transformers: State-of-the-art natural language processing. In Q. Liu & D. Schlangen (Eds.), *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations* (pp. 38–45). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2020.emnlp-demos.6>
- Zarrieß, S., Voigt, H., & Schüz, S. (2021). Decoding methods in neural language generation: A survey. *Information*, 12(9), 355. <https://doi.org/10.3390/info12090355>
- Zhang, H., Song, H., Li, S., Zhou, M., & Song, D. (2022). A survey of controllable text generation using transformer-based pre-trained language models. *arXiv preprint arXiv:2201.05337*. <https://doi.org/10.48550/arXiv.2201.05337>
- Zhou, S., Liu, J., Zhong, X., & Zhao, W. (2021). Named entity recognition using bert with whole world masking in cybersecurity domain. *2021 IEEE 6th International Conference on Big Data Analytics (ICBDA)*, 316–320. <https://doi.org/10.1109/ICBDA51983.2021.9403180>
- Zhu, L., Xu, Y., Zhu, Z., Bao, Y., & Kong, X. (2022). Fine-grained sentiment-controlled text generation approach based on pre-trained language model. *Applied Sciences*, 13(1), 264. <https://doi.org/10.48550/arXiv.2006.09891>
- Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., & Irving, G. (2019). Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*. <https://doi.org/10.48550/arXiv.1909.08593>

## Declaration of Originality

I hereby confirm that I

- have written this Thesis independently and without the help of any third party,
- have provided all the sources and cited all the literature I used,
- will protect the confidentiality of the client and respect the copyright regulations of Lucerne University of Applied Sciences and Arts.

Date and signature:



Paulina Aleksandra Żal

Wettingen AG, 22nd of December 2023

Lucerne University of Applied Sciences and Arts | (HSLU)

Master of Science in Applied Information and Data Science (MScIDS)

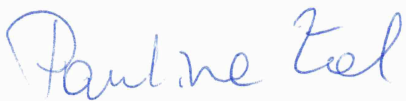


## Declaration of the use of Generative AI

For the work in question, I have used Generative AI to:

- ☒ brainstorm new ideas
- ☐ generate new research hypotheses
- ☐ do scientific research
- ☒ summarize other research
- ☐ set up the research/project design
- ☒ code
- ☐ write the manuscript
- ☒ edit or translate the text

Date and signature:



Paulina Aleksandra Żal

Wettingen AG, 22nd of December 2023

Lucerne University of Applied Sciences and Arts | (HSLU)

Master of Science in Applied Information and Data Science (MScIDS)