Overcoming Bureaucratic Exclusion Addressing the Challenges Faced by Foreigners in Germany

Franz Christopher Xaver Hagen
Universitat der Kunste
MA Design & Computation
fr.hagen@udk-berlin.de

Abstract

This paper introduces a novel framework for easing the transition into new countries, specifically targeting the challenges posed by navigating complex bureaucratic systems in a foreign language. The proposed Flask-based web application leverages Optical Character Recognition (OCR), Natural Language Processing (NLP), machine translation, and a custom-developed language model to demystify and streamline intricate German bureaucratic documents, making them comprehensible and accessible. Central to our approach is the application of the Direct Preference Optimization (DPO) method in the development of a language model specifically tailored to simplify complex texts while ensuring the preservation of essential information. By training this model on a diverse dataset derived from Klexikon, we have enabled it to effectively distill dense bureaucratic jargon into clear, accessible language. This paper focuses on the optimal integration of these technological components on a comprehensive scale, exemplified through the development process and the application's capacity to significantly enhance the clarity of bureaucratic documentation for non-German speakers.

I. Introduction

The integration of immigrants into the social fabric of Germany presents an array of challenges, exacerbated by the complexities of bureaucratic structures. This phenomenon has become particularly pronounced in the context of increasing globalization and migration flows, imposing significant barriers for non-German speakers 1. These barriers often manifest in the form of intricate bureaucratic texts that impede access to vital information and, consequently, hinder social integration [6]. Although extant literature has delved into critical aspects of integration such as language acquisition and labor market entry, the potential of technology to surmount the linguistic hurdles of bureaucracy is not yet fully realized.

To address the identified challenge, this study introduces a web-based application equipped with a specialized language model utilizing Direct Preference Optimization (DPO) technology [8]. This innovation is designed to enhance the comprehensibility of German bureaucratic documents, thereby facilitating the bureaucratic integration process for immigrants. The application employs Simple German, a variant of language known for its plain or easy-to-read characteristics, distinguished by reduced lexical and syntactic complexity. This language variant is instrumental in making information accessible to a broad spectrum of individuals, including those with cognitive disabilities like aphasia and dyslexia, foreign language learners, Deaf individuals, and children [4]. Utilizing DPO [8], the application balances the need for clarity with the preservation of essential information, drawing upon an extensive Klexikon dataset [1] that comprises over 7.6 million characters of simplified text tailored to the inclusivity needs of bureaucratic communication.

I evaluate our model's effectiveness against existing simplification strategies through a com-

prehensive framework, employing both quantitative and qualitative metrics. This evaluation seeks to determine the most effective model for making bureaucratic language understandable while preserving essential information. Our approach, grounded in a bespoke dataset tailored for simplification, represents a novel contribution to the field of bureaucratic document simplification.

By merging the DPO [8] technique with this specialized dataset, I aim to enhance bureaucratic accessibility and inclusivity, addressing the needs of those disadvantaged by language barriers. This endeavor contributes to a broader discourse on societal inclusivity, emphasizing the importance of accessible communication within diverse populations.

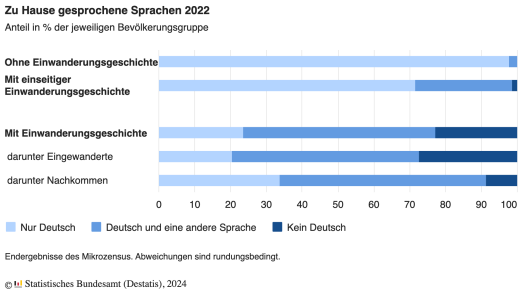

Figure 1: *The chart shows the linguistic composition of German households in 2024, differentiated according to the proportion of people with and without a migration background and their language skills. It can be seen that the majority of people without a migration background speak only German, while the groups with a migration background are multilingual. The data source is the Federal Statistical Office for the year 2024.*`https://www.destatis.de/DE/Presse/Pressemitteilungen/2024/02/PD24_N008_12.html`

Drawing inspiration from the successes of contemporaneous scholars who have harnessed technological interventions to address a spectrum of issues ranging from the enhancement of financial planning for students [9] to the conservation of endangered languages and cultures [5]. This investigation contributes to the ongoing discourse on social inclusivity. It emphasizes the imperative of accessible communication within bureaucratic systems for diverse demographics.

I suggest a web-based system that allows users to upload images of their bureaucratic documents. By utilizing Optical Character Recognition (OCR) [1] technology, the system extracts the text and, leveraging a custom language model, translates it into a simplified version that is easier to comprehend. By creating an accessible and effective text simplification tool, we aim to improve text comprehensibility for all user groups, thereby contributing to the social integration of non-German speaking immigrants in Germany.

II. State of the art

This research endeavors to address the critical need for technological advancements in facilitating the bureaucratic integration of immigrants, thus aiming to bridge a notable void within the existing scholarly milieu. Through a meticulous examination of the prevailing landscape in language simplification and classification, this study offers a comprehensive critique of established methodologies, including the competency levels defined by the Council of Europe and the Lexile scale [6]. An in-depth analysis of texts conforming to the tenets of "Leichte" and "Einfache Sprache" [7] reveals the extant practices in inclusive language, identifying a distinct lack of thorough and accessible datasets for Simple German [10] , with a particular focus on materials for individuals with migration backgrounds.

In an endeavor to pioneer novel methodologies, this study introduces an innovative text simplification strategy anchored in direct preference optimization [8], subject to rigorous evaluation to verify its efficacy. This approach is designed to improve the clarity of text for a wide array

[1] `https://github.com/tesseract-ocr/tesseract`

of user demographics, thereby significantly enhancing the social integration of non-German speaking immigrants in Germany.

Additionally, this paper critically examines contemporary systems, such as the one available at[2], which predominantly utilizes artificial intelligence (AI). Despite the substantial advantages AI provides in data processing and analysis, its excessive dependence without adequate human supervision can lead to critical inaccuracies or misinformation. This concern accentuates the significance of our research, which not only employs sophisticated computational techniques but also ensures these methods undergo comprehensive evaluation and are anchored in principles centered around human oversight. Consequently, this research aims to develop a more dependable and efficacious instrument for text simplification, thereby contributing to the overarching dialogue on the role of technology in promoting more inclusive and precise communication within bureaucratic contexts.

III. Data

In this research, we engaged with the Klexikon dataset, sourced from the German children's encyclopedia known as "Klexikon," [3] which is renowned for its succinct and intelligible content designed for children aged 8 to 13 years. The dataset, crafted by Dennis Aumiller and Michael Gertz [2], was meticulously compiled through a process of aligning entries from Klexikon with corresponding articles from the German Wikipedia. This alignment was selective, considering only Wikipedia articles of more than 15 paragraphs to avoid ambiguities and ensure comprehensiveness. This procedure revealed a stark disparity in content length, with Wikipedia articles exhibiting an average sentence count nearly ninefold that of their Klexikon counterparts 1. Consisting of 2,898 matched articles, the dataset provides a solid foundation for training our language model, effectively bridging the divide between sophisticated bureaucratic jargon and simplified text. This endeavor aims to improve the accessibility of texts for non-German speakers dealing with the German bureaucratic system. Further enhancing our dataset's quality, we employed regular expressions to refine the data cleaning process, thereby augmenting the utility of the data for our training model and ensuring the analytical accuracy and reliability required for financial analysis. This strategy offers a structured approach to data normalization, enabling a streamlined and more error-resistant preparation stage for the model's training phase.

Table 1: *Corpus statistics of the Klexikon dataset. SD refers to one standard deviation.*

	Wikipedia	**Klexikon**
Documents	2,898	2,898
Average sentences	242.09	32.51
SD sentences	227.39	19.73
Median sentences	162	26
Average tokens	5,442.83	436.87
SD tokens	5,093.82	270.00
Median tokens	3,705	347

IV. Optimization method(s)

This study introduces a Flask-based web application that incorporates Optical Character Recognition (OCR), Natural Language Processing (NLP), machine translation, and a custom language model to render complex texts more accessible to a diverse readership.

Central to our approach is the newly compiled Klexikon dataset [1], based on the German children's encyclopedia "Klexikon"[4]. Selected for its high-quality, simplified content this dataset provides a stable foundation for model training. Our methodology employs Direct Preference Optimization (DPO) [8], an innovative policy

[2] https://eye-able.com/audit/wcagtest

[3] https://klexikon.zum.de/wiki/Klexikon:Willkommen_im_Klexikon

[4] https://klexikon.zum.de/wiki/Klexikon:Willkommen_im_Klexikon

optimization technique guided by direct user preferences. Distinct from traditional methods that optimize an explicit reward function via reinforcement learning, DPO [8] trains the language model directly based on human feedback. This negates the need for a reward model by converting the traditional loss—measured as the difference between the expected and received reward—into a loss based on the divergence between the preferred and model-generated actions.

The formula

$$\pi_r(y|x) = \frac{1}{Z(x)} \pi_{ref}(y|x) \exp\left(\frac{1}{\beta} r(x,y)\right)$$

demonstrates how the optimal policy $\pi_r(y|x)$ is derived from the reference policy $\pi_{ref}(y|x)$ and the reward function $r(x,y)$, modulated by the factor β. Here $Z(x)$ functions as a normalizing factor, ensuring probability distribution across all possible actions. Our findings substantiate the efficacy of DPO in training language models to convert complex texts into simplified content that is intelligible to a wider audience [8]. This method is particularly advantageous for fine-tuning language models by directly catering to the linguistic needs of children and individuals with varied levels of language proficiency, as well as by bypassing the complexities associated with conventional Reinforcement Learning (RL)[5] techniques.

[5]Reinforcement Learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment to maximize the notion of cumulative reward.

I. Model Testing

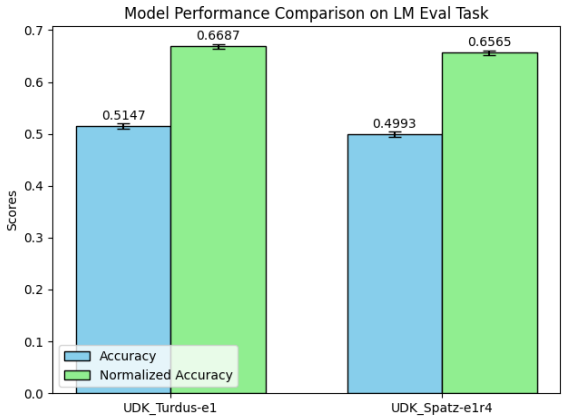

Figure 2: *The figure illustrates a performance comparison between two language models, UDK_Turdus-e1 and UDK_Spatz-e1r4, based on two metrics: Accuracy and normalized accuracy. The bars show the scores achieved by both models, with the UDK_Turdus-e1 model showing higher normalized accuracy, while UDK_Spatz-e1r4 lags slightly behind in the accuracy category. The error bars indicate the variability of the results.*

Table 2: *This is a table. The data shows the results for the original model UDK_Turdus-e1 and the model UDK_Spatz-e1r4.*

Tasks Metric	Version Value	Filter	n-shot Stderr
UDK_Turdus-e1			
hellaswag_de acc	1 0.5147	none ±	5 0.0052
acc_norm	0.6687	none ±	5 0.0049
UDK_Spatz-e1r4			
hellaswag_de acc	1 0.4993	none ±	5 0.0052
acc_norm	0.6565	none ±	5 0.0049

This research entailed using the lm-evaluation-harness framework [3] provided by EleutherAI [3] to evaluate the response accuracy and confidence of two iterations of a language model: the baseline "UDK_Turdus-e1" [11] and its refined version "UDK_Spatz-e1r4." The goal was to analyze the effects of fine-tuning on the model's performance and to assess the dependability of its responses.

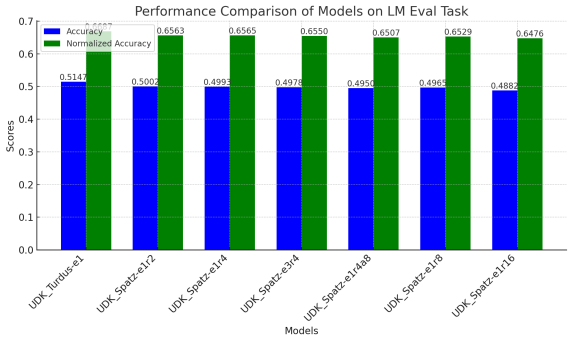

Figure 3: *Comparison of the performance of models in the LM Eval task*

For this evaluation, we selected the "hellaswag_de" task [3], a German-adapted version of the well-known HellaSWAG task, which tests the models' comprehension of contexts and their ability to predict logical continuations. The task was conducted without specific filters ("none"), employing a 5-shot learning approach, wherein five exemplar texts were presented to the models prior to the actual evaluation. Model performance was assessed using two metrics: accuracy (*acc*) and normalized accuracy (*acc_norm*), to glean a comprehensive picture of response quality. The results indicated an accuracy of 0.5147 and a normalized accuracy of 0.6687 for the original "UDK_Turdus-e1" model. The fine-tuned "UDK_Spatz-e1r4" model achieved an accuracy of 0.4993 and a normalized accuracy of 0.6565. Both models exhibited similar standard errors, suggesting a consistent evaluation method. The observed marginal decrease in accuracy for the fine-tuned model may be attributed to specific adjustments made to enhance response confidence. These findings suggest that fine-tuning for increased certainty may require minor compromises in accuracy 3. Nonetheless, the performance remains within an acceptable range, affirming the reliability and precision of the fine-tuned model for practical applications.

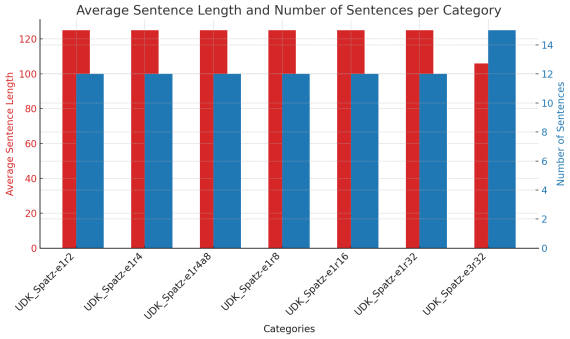

Figure 4: *The graph compares the average sentence length and sentence count across categories for UDK_Spatz model variants. Models UDK_Spatz-e1r2 through UDK_Spatz-e3r32 are arrayed on the x-axis, with average sentence length indicated by red bars and sentence count by blue bars on dual y-axes. The UDK_Spatz-e3r32 stands out with the highest sentence length and count, and error bars signal the variability of the average sentence lengths.*

II. Evaluation by people

This investigation aimed to ascertain the efficacy of a text simplification website in augmenting the comprehensibility of German language texts for individuals for whom German is not their native tongue. Ten participants were enlisted to evaluate the clarity of specific texts 8 before and after the application of text simplification techniques. Initially, each participant rated the understandability of a given text by choosing one of three descriptors: "Understood Nothing," "Understood a Little," or "Understood Everything." Thereafter, the text was processed through the text simplification website (`http://194.95.202.207:5000/`) and subjected to a second assessment.

Prior to the application of text simplification,

seven participants indicated a complete lack of understanding, three acknowledged partial comprehension, and none affirmed full understanding. Following the intervention of the text simplification tool, the number of participants who achieved complete understanding rose to five, with three retaining partial comprehension and two continuing to express no understanding. The results indicate a significant enhancement in text comprehensibility attributable to the text simplification website for individuals learning German as a second language. This study observes a noteworthy increment in participants' complete comprehension post-processing with the simplification software. Further exploration may refine the utility of such tools and elucidate the text simplification elements that most effectively bolster understandability.

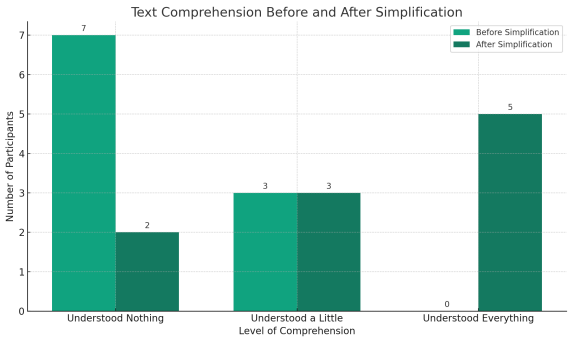

Figure 5: *the study shows the changes in text comprehensibility on the basis of 10 test subjects.*

V. Result

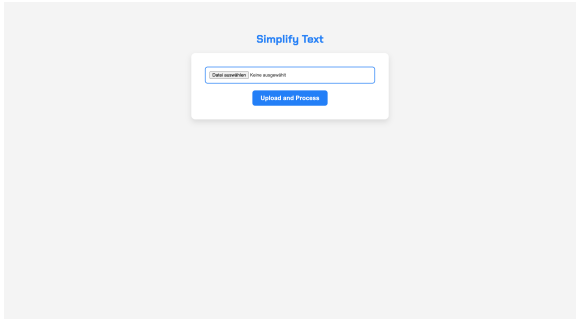

Figure 6: *Screenshot of my website where it is possible to upload files (`http://194.95.202.207:5000/`)*

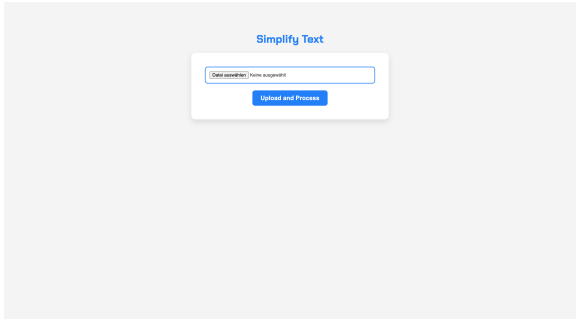

Figure 7: *Screenshot from the website showing what is displayed when the simplification process is finished(`http://194.95.202.207:5000/`)*

During the development of our web application for simplifying bureaucratic texts, we encountered several challenges. One of the most significant was the incorporation of a custom language model designed to convert complex sentences into more comprehensible language while retaining the original meaning. Through the analysis of sentence structures and fine-tuning of model parameters, we succeeded in training the language model to generate concise and clear responses.

To assess the effectiveness of the language model, we conducted a study with ten participants who speak German as a second language. The purpose of this study was to evaluate the understandability of the texts produced

by our language model. Participant feedback provided insights into the text simplification's efficacy, prompting further refinements to the model.

The study's findings indicated a substantial improvement in text comprehensibility for individuals learning German as a second language. Participants affirmed the language model's effectiveness, corroborating our hypothesis that a specially tailored language model can enhance the accessibility of complex bureaucratic texts.

In order to ensure that the adapted language model did not deviate excessively from the original model, it was further evaluated using EleutherAI's LM Evaluation Harness [3]. This testing confirmed that the model maintained its general performance capabilities despite its specialization in generating succinct and straightforward responses.

VI. Discussion

The development of our web application is directed toward facilitating the comprehension and integration of non-German-speaking individuals into bureaucratic processes. By integrating a bespoke language model, specifically optimized for the simplification of complex texts, we have achieved significant advancements in making such documents more accessible. Our study involved ten participants who speak German as a second language, evaluating the effectiveness and comprehensibility of the language model. The results unequivocally indicate that the web application and the integrated language model effectively enhance the understanding of bureaucratic texts 5. This was accomplished through targeted analysis and adaptation of the language model, with an emphasis on brief and clear responses. Participant feedback corroborated the efficacy of the texts simplified by the language model, thereby confirming our hypothesis.

The application of the Direct Preference Optimization (DPO) [8] approach, followed by validation through the LM Evaluation Harness [3], confirmed the generalizability and quality of the text simplification, without diverging substantially from the original language model. Direct optimization of preferences has proven to be an effective method for tailoring language models to specific requirements.

However, limitations regarding the validity, reliability, and generalizability of our findings persist. Future research should focus on expanding the study to include a larger and more diverse pool of participants, as well as ongoing refinement and enhancement of the language model based on extensive data and feedback.

Our research presents a practical exemplar of the application of language models for text simplification, demonstrating how technical solutions can be employed to dismantle linguistic barriers and foster integration. The positive feedback and the outcomes achieved reinforce our conviction that such technological approaches can significantly contribute to overcoming language and cultural barriers.

VII. Conclusion

The present study delineates the creation of a web application aimed at ameliorating the linguistic and bureaucratic hurdles encountered by individuals who do not speak German as their first language. By harnessing state-of-the-art technologies such as Optical Character Recognition (OCR), Natural Language Processing (NLP), machine translation, and a custom language model honed via Direct Preference Optimization (DPO) [8], the application renders complex bureaucratic texts into comprehensible content. This initiative bridges a crucial gap by furnishing tools that enhance clarity in bureaucratic communication, thus aiding the integration of immigrants.

Our investigation reveals that the newly developed web application and its specialized

language model substantially enhance the intelligibility of intricate bureaucratic documents. Utilizing DPO 4, the model is crafted to distill content into its essence, ensuring the preservation of vital information. The efficacy of the system is corroborated by affirmative feedback from participants and the application's validation through the LM Evaluation Harness [3] platform, confirming its utility and dependability.

The repercussions of this research span a wide range, proposing a methodology to streamline the bureaucratic assimilation of immigrants and to elevate the accessibility of information. It posits that ongoing investigation is imperative to refine the model's precision and to broaden its linguistic proficiency, thus guaranteeing its extensive application and a more inclusive user experience.

This web application constitutes a pivotal advancement in surmounting linguistic obstacles and fostering inclusivity within bureaucratic frameworks. Subsequent efforts should concentrate on enhancing the generalizability and adaptability to various user preferences, with the aim of amplifying the reach and efficacy of such technological interventions. The current study exemplifies the potential of language technologies to forge connections across linguistic and cultural divides, thereby supporting integrative practices in an increasingly globalized context.

VIII. Where do You go from here ?

This research lays the groundwork for an in-depth exploration of a web application's enduring influence on non-German speakers' bureaucratic integration. An analysis of sustained user feedback and the application's long-term efficacy will yield significant insights into performance and areas for improvement.

The study acknowledges the necessity of a diverse participant cohort to genuinely represent the experiences of non-German speakers in Germany. An expansion of the participant demographic promises to bolster the research findings and to enhance the application's adaptability and utility across various user groups.

Moreover, the study proposes the development of an intuitive application that enables straightforward document uploads or image captures through smartphones. Such an advancement would significantly enhance user engagement and convenience, offering an on-the-go solution for document processing.

The integration of Optical Character Recognition (OCR) technology is crucial in accurately determining the layout of textual elements within documents, thereby assisting users in efficiently navigating and locating pertinent information.

Furthermore, the application's capacity for handling large PDF files is anticipated to be a considerable improvement, ensuring users can comprehensively process extensive documents within the system, which is essential for a detailed bureaucratic assessment.

By focusing on these aspects, the application will transcend its current functionality to become an expansive, user-oriented tool. It will address a wider spectrum of user requirements, thus providing a more profound understanding of its long-term utility and effectiveness.

IX. Acknowledgments

I am thankful to Prof. Dr. Daniel D. Hromada for his invaluable guidance, knowledge, and support, all of which played a crucial role in the successful completion of this project. I also want to express my gratitude to my peers in the Design & Computation [6] program for their camaraderie, insightful exchanges, and steadfast support during our collective endeavor to navigate the complexities of "Wicked Solutions."

[6] https://www.newpractice.net/study

The project greatly benefited from this collaborative effort, enhancing both its process and final result.

References

[1] Dennis Aumiller and Michael Gertz. *Klexikon: A German Dataset for Joint Summarization and Simplification*. 2022. arXiv: 2201.07198 [cs.CL].

[2] Dennis Aumiller and Michael Gertz. "Klexikon: A German Dataset for Joint Summarization and Simplification". In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, June 2022, pp. 2693–2701. URL: https://aclanthology.org/2022.lrec-1.288.

[3] Leo Gao et al. *A framework for few-shot language model evaluation*. Version v0.4.0. Dec. 2023. DOI: 10.5281/zenodo.10256836. URL: https://zenodo.org/records/10256836.

[4] David Klaper, Sarah Ebling, and Martin Volk. "Building a German/simple German parallel corpus for automatic text simplification". In: (2013).

[5] Lilli-Chiara Kurth. "rěc :: Language Learning System :: Lower Sorbian". In: *Wicked Solutions* 0 (2024).

[6] Christiane Maaß. *Easy language–plain language–easy language plus: Balancing comprehensibility and acceptability*. Frank & Timme, 2020.

[7] Christiane Maaß. *Leichte Sprache. Das Regelbuch.* Lit-Verlag, 2015.

[8] Rafael Rafailov et al. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. 2023. arXiv: 2305.18290 [cs.LG].

[9] Pierre-Louis Wolfgang Léon Suckrow. "Navigating (personal) finances in a latent way". In: *Wicked Solutions* 0 (2024).

[10] Vanessa Toborek et al. *A New Aligned Simple German Corpus*. 2023. arXiv: 2209.01106 [cs.CL].

[11] UDK dot AI, Daniel Devatman Hromada. *Turdus (Revision 923c305)*. 2024. DOI: 10.57967/hf/1611. URL: https://huggingface.co/udkai/Turdus.

X. Appendix

Model	Tasks	Version	Filter	n-shot	Metric	Value	Stderr
UDK_Turdus-e1	hellaswag_de	1	none	5	acc	0.5147	± 0.0052
UDK_Turdus-e1	hellaswag_de	1	none	5	acc_norm	0.6687	± 0.0049
UDK_Spatz-e1r2	hellaswag_de	1	none	5	acc	0.5002	± 0.0052
UDK_Spatz-e1r2	hellaswag_de	1	none	5	acc_norm	0.6563	± 0.0049
UDK_Spatz-e1r4	hellaswag_de	1	none	5	acc	0.4993	± 0.0052
UDK_Spatz-e1r4	hellaswag_de	1	none	5	acc_norm	0.6565	± 0.0049
UDK_Spatz-e3r4	hellaswag_de	1	none	5	acc	0.4978	± 0.0052
UDK_Spatz-e3r4	hellaswag_de	1	none	5	acc_norm	0.6550	± 0.0049
UDK_Spatz-e1r4a8	hellaswag_de	1	none	5	acc	0.4950	± 0.0052
UDK_Spatz-e1r4a8	hellaswag_de	1	none	5	acc_norm	0.6507	± 0.0049
UDK_Spatz-e1r8	hellaswag_de	1	none	5	acc	0.4965	± 0.0052
UDK_Spatz-e1r8	hellaswag_de	1	none	5	acc_norm	0.6529	± 0.0049
UDK_Spatz-e1r16	hellaswag_de	1	none	5	acc	0.4882	± 0.0052
UDK_Spatz-e1r16	hellaswag_de	1	none	5	acc_norm	0.6476	± 0.0049

Table 3: *Comparison of model performance metrics across various versions*

§ 271a Vereinbarungen über Zahlungs-, Überprüfungs- oder Abnahmefristen

(1) Eine Vereinbarung, nach der der Gläubiger die Erfüllung einer Entgeltforderung erst nach mehr als 60 Tagen nach Empfang der Gegenleistung verlangen kann, ist nur wirksam, wenn sie ausdrücklich getroffen und im Hinblick auf die Belange des Gläubigers nicht grob unbillig ist. Geht dem Schuldner nach Empfang der Gegenleistung eine Rechnung oder gleichwertige Zahlungsaufstellung zu, tritt der Zeitpunkt des Zugangs dieser Rechnung oder Zahlungsaufstellung an die Stelle des in Satz 1 genannten Zeitpunkts des Empfangs der Gegenleistung. Es wird bis zum Beweis eines anderen Zeitpunkts vermutet, dass der Zeitpunkt des Zugangs der Rechnung oder Zahlungsaufstellung auf den Zeitpunkt des Empfangs der Gegenleistung fällt; hat der Gläubiger einen späteren Zeitpunkt benannt, so tritt dieser an die Stelle des Zeitpunkts des Empfangs der Gegenleistung. (2) Ist der Schuldner ein öffentlicher Auftraggeber im Sinne von § 99 Nummer 1 bis 3 des Gesetzes gegen Wettbewerbsbeschränkungen, so ist abweichend von Absatz 1

1. eine Vereinbarung, nach der der Gläubiger die Erfüllung einer Entgeltforderung erst nach mehr als 30 Tagen nach Empfang der Gegenleistung verlangen kann, nur wirksam, wenn die Vereinbarung ausdrücklich getroffen und aufgrund der besonderen Natur oder der Merkmale des Schuldverhältnisses sachlich gerechtfertigt ist;

2. eine Vereinbarung, nach der der Gläubiger die Erfüllung einer Entgeltforderung erst nach mehr als 60 Tagen nach Empfang der Gegenleistung verlangen kann, unwirksam.

Absatz 1 Satz 2 und 3 ist entsprechend anzuwenden.

(3) Ist eine Entgeltforderung erst nach Überprüfung oder Abnahme der Gegenleistung zu erfüllen, so ist eine Vereinbarung, nach der die Zeit für die Überprüfung oder Abnahme der Gegenleistung mehr als 30 Tage nach Empfang der Gegenleistung beträgt, nur wirksam, wenn sie ausdrücklich getroffen und im Hinblick auf die Belange des Gläubigers nicht grob unbillig ist.

(4) Ist eine Vereinbarung nach den Absätzen 1 bis 3 unwirksam, bleibt der Vertrag im Übrigen wirksam.

(5) Die Absätze 1 bis 3 sind nicht anzuwenden auf

1. die Vereinbarung von Abschlagszahlungen und sonstigen Ratenzahlungen sowie
2. ein Schuldverhältnis, aus dem ein Verbraucher die Erfüllung der Entgeltforderung schuldet.

(6) Die Absätze 1 bis 3 lassen sonstige Vorschriften, aus denen sich Beschränkungen für Vereinbarungen über Zahlungs-, Überprüfungs- oder Abnahmefristen ergeben, unberührt.

Figure 8: *Test segment wich was used in the study. This segment of the study examines participant comprehension of a specific clause within the German Civil Code (Bürgerliches Gesetzbuch, BGB* `https://beck-online.beck.de/Print/CurrentDoc?vpath=bibdata/komm/daulankobgb_4/bgb/cont/daulankobgb.bgb.p271a.htm&printdialogmode=CurrentDoc&hlword=`*, which is available for detailed review in the database of Beck-Online.*

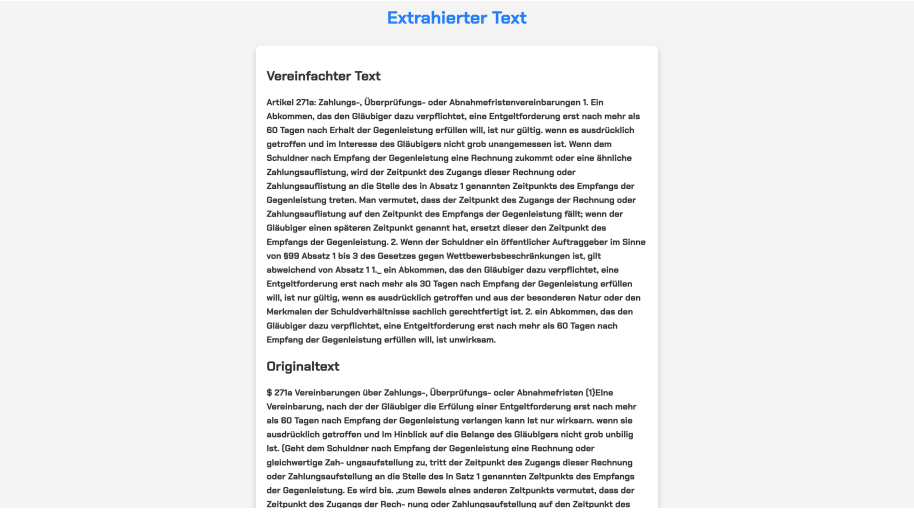

Figure 9: *Screenshot from the website showing what is displayed when the simplification process is finished*

```python
from flask import Flask, jsonify, request, render_template
from flask_cors import CORS
from werkzeug.utils import secure_filename
import os
from dotenv import load_dotenv
from ocr_utils import extract_text_from_image
from api_utils import simplify_text

UPLOAD_FOLDER = '/home/franz/Simple_Web/UPLOAD_FOLDER'

# App configuration and initialization
app = Flask(__name__)
CORS(app)
load_dotenv()
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
allowed_extensions = {'png', 'jpg', 'jpeg', 'pdf'}

# API key and other environment variables
openai_api_key = os.getenv("OPENAI_API_KEY")

@app.route('/', methods=['POST'])
def process_image():
    file = request.files.get('file')
    if file and allowed_file(file.filename):
        extracted_text = extract_text_from_image(file.stream)
        if extracted_text:
            simplified_text = simplify_text(extracted_text)
            return render_template('result.html', simplified_text=
                simplified_text,original_text=extracted_text)
            return jsonify({'message': 'Upload successful', 'data': 'Your
                processed data here'})

        results_url = url_for('results', filename=filename, _external=True)
        return jsonify({'results_url': results_url}), 200
        return jsonify({'error': 'Text extraction failed'}), 500
    return jsonify({'error': 'No file part or invalid file type'}), 400

@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
        allowed_extensions

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

Listing 1: *Python Code for Flask app for processing images and simplifying text*

```python
import pytesseract
from PIL import Image, ImageEnhance, ImageFilter
import logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

def extract_text_from_image(image_path, lang='deu', preprocessing='thresh'):
    try:
        with Image.open(image_path) as img:
            if preprocessing == 'thresh':
                img = img.convert('L')  # Convert to grayscale
                img = img.point(lambda x: 0 if x < 128 else 255, '1')
            elif preprocessing == 'enhance':
                enhancer = ImageEnhance.Contrast(img)
                img = enhancer.enhance(2)
            text = pytesseract.image_to_string(img, lang=lang)
        return text
    except FileNotFoundError as e:
        logging.error(f"File not found: {e}")
        return None
    except IOError as e:
        logging.error(f"Error opening or processing image: {e}")
        return None
```

Listing 2: *Python Code for OCR text extraction from image*

```python
import os
from openai import OpenAI
import re

client = OpenAI(
    api_key="XXXXXXXXXXXX",
    base_url="https://XXXXXXXX/XXX/",
)

def simplify_text(text):
    prompt = "Hier ist ein schwieriger text:\n\n"
    prompt += text
    prompt += "\n\n Hier ist der gleiche Text in einfacher Sprache:\n\n"

    chat_completion = client.completions.create(
        prompt=prompt,
        max_tokens=512,
        temperature=0.7,
        frequency_penalty=0.0,
        presence_penalty=0.0,
        top_p=0.9,
        seed=42,
        model="UDK_Spatz-e1r4"
    )

    response_text = chat_completion.choices[0].text.strip()

    # Post-processing to ensure ending with a complete sentence
    # This is a basic approach and might need refinement for better accuracy
    if not response_text.endswith('.'):
        response_text = '.'.join(response_text.split('.')[:-1]) + '.'

    text = response_text
    saetze = re.split(r'[.!?;]', text)

    # Leere S tze entfernen und Whitespaces trimmen
    saetze = [satz.strip() for satz in saetze if satz.strip()]

    # Durchschnittliche L nge der S tze berechnen
    durchschnittliche_laenge = sum(len(satz) for satz in saetze) / len(saetze)

    # Anzahl der S tze und durchschnittliche L nge ausgeben
    anzahl_saetze = len(saetze)
    print(f"Durchschnittliche L nge der S tze: {durchschnittliche_laenge}, Anzahl der S tze: {anzahl_saetze}")


    return response_text
```

Listing 3: *Python code for the API Callback*

```
from unsloth import FastMistralModel, PatchDPOTrainer
from peft import LoraConfig, PeftModel, get_peft_model,
    prepare_model_for_kbit_training
import torch
from datasets import Dataset, load_dataset
from dataclasses import dataclass, field
from typing import Dict, Optional
import json
from transformers import TrainingArguments
from trl import DPOTrainer

epochs = 3
rank = 4

model_name = "/data/LLM/models/UDK_Turdus-e1"
new_model = "/data/LLM/models/UDK_Spatz-e" + str(epochs) + "r" + str(rank)
dataset = load_dataset("json", data_files="klexikon_dpo_all.json")['train']

max_seq_length = 255
dtype = None

def substitute_proper_nouns(entry):
    chosen = entry["chosen"]
    reject = entry["reject"]

    if chosen[0].isupper() and reject[0].isupper():
        entry["chosen"] = chosen[:-1]
        entry["reject"] = reject[:-1]
        entry["prompt"] = entry["prompt"].replace(chosen,chosen[:-1]).
            replace(reject,reject[:-1])

    return entry

def extract_anthropic_prompt(prompt_and_response):
    """Extract the anthropic prompt from a prompt and response pair."""
    search_term = "\n\nAssistant:"
    search_term_idx = prompt_and_response.rfind(search_term)
    assert search_term_idx != -1, f"Prompt and response does not contain '{
        search_term}'"
    return prompt_and_response[: search_term_idx + len(search_term)]

def get_hh(split: str, sanity_check: bool = False, silent: bool = False,
    cache_dir: str = None) -> Dataset:
    """Load the Anthropic Helpful-Harmless dataset from Hugging Face and
        convert it to the necessary format.

    The dataset is converted to a dictionary with the following structure:
    {
        'prompt': List[str],
        'chosen': List[str],
        'rejected': List[str],
    }
```

```
48
49      Prompts should be structured as follows:
50        \n\nHuman: <prompt>\n\nAssistant:
51      Multiple turns are allowed, but the prompt should always start with \n\
            nHuman: and end with \n\nAssistant:.
52      """
53      dataset = load_dataset("Anthropic/hh-rlhf", split=split, cache_dir=
            cache_dir)
54      if sanity_check:
55          dataset = dataset.select(range(min(len(dataset), 1000)))
56
57      def split_prompt_and_responses(sample) -> Dict[str, str]:
58          prompt = extract_anthropic_prompt(sample["chosen"])
59          return {
60              "prompt": prompt,
61              "chosen": sample["chosen"][len(prompt) :],
62              "rejected": sample["rejected"][len(prompt) :],
63          }
64
65      return dataset.map(split_prompt_and_responses)
66
67  def chatml_format(example):
68      example=substitute_proper_nouns(example)
69      parts=example['prompt'].split('_')
70      return {
71          "prompt": parts[0],
72          "chosen": example['chosen']+parts[1],
73          "rejected": example['reject']+parts[1],
74      }
75
76  original_columns = dataset.column_names
77
78  train_dataset = dataset
79
80  model, tokenizer = FastMistralModel.from_pretrained(
81      model_name = model_name,
82      max_seq_length = max_seq_length,
83      dtype = dtype,
84      load_in_4bit = False
85  )
86
87  print("LOADED " + model_name)
88
89  model = FastMistralModel.get_peft_model(
90      model,
91      r = rank,
92      target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
93                        "gate_proj", "up_proj", "down_proj",],
94      lora_alpha = 4,
95      lora_dropout = 0, # Currently only supports dropout = 0
96      bias = "none",    # Currently only supports bias = "none"
97      use_gradient_checkpointing = True,
```

```
    random_state = 42,
    max_seq_length = max_seq_length,
)

dpo_trainer = DPOTrainer(
    model,
    ref_model=None,
    args = TrainingArguments(
        per_device_train_batch_size = 4,
        gradient_accumulation_steps = 4,
        lr_scheduler_type="cosine",
        warmup_ratio = 0.1,
        learning_rate=5e-5,
        warmup_steps=100,
        num_train_epochs = epochs,
        fp16 = not torch.cuda.is_bf16_supported(),
        bf16 = torch.cuda.is_bf16_supported(),
        logging_steps = 1,
        optim = "paged_adamw_32bit",
        seed = 42,
        output_dir = "outputs",
    ),
    beta=0.1,
    train_dataset=train_dataset,
    tokenizer=tokenizer,
)

dpo_trainer.train()
dpo_trainer.model.save_pretrained("final_checkpoint")
tokenizer.save_pretrained("final_checkpoint")

adapter_model = dpo_trainer.model.merge_and_unload()
adapter_model.save_pretrained(new_model)
tokenizer.save_pretrained(new_model)
```

Listing 4: *Python code used for DPO training of the Language Models,*