

Research Paper Series

Nr. 14 | Februar 2024

Classification of Intent in Moderating Online Discussions: An Empirical Evaluation

Autoren:
Christoph Gehweiler, Oleg Lobachev

Autoren:

Christoph Gehweiler

Leibniz-Fachhochschule

Oleg Lobachev

Leibniz-Fachhochschule

oleg.lobachev@leibniz-fh.de

ISSN 2511-7491

Redaktion: Prof. Dr. Robin Christmann

Classification of Intent in Moderating Online Discussions: An Empirical Evaluation

CHRISTOPHER GEHWEILER

OLEG LOBACHEW

Zusammenfassung / Abstract:

This paper investigates the use of large language models (LLMs) for moderating online discussions, with a focus on identifying user intent in various types of content. It centers on natural language processing (NLP) techniques to detect toxic language, derailment in discussions, and problematic comments. The study creates prototypes of such tools, utilizing LLMs. Then we evaluate our tools using datasets in both English and German, as the effectiveness across different languages may vary. This research explores content classification through methods such as sentiment analysis, keyword extraction and topic modeling, employing non-binary labeling for a deeper analysis of online interactions. The paper also discusses the limitations of current LLMs, including the challenge of false positives due to limited training data. It concludes with ideas towards improving model fine-tuning to better address specific platform needs and linguistic variations. This work contributes to understanding how AI can support decisions in moderating online spaces and fostering healthier digital communication environments.

Fachrichtung / Field of Study: Computer Science

Schlagworte / Keywords: natural language processing, machine learning, artificial intelligence, BERT, discussion, moderation, sentiment analysis, spam, toxicity, language, decision support

1. Introduction

The field of artificial intelligence (AI), particularly natural language processing (NLP), is pivotal in the classification of intent. Our study utilizes the concept of AI-assisted moderation in online discussions as a practical example to demonstrate the latest NLP techniques. These methods are explored as potential aids for moderators in making informed decisions. This paper is based on first author's thesis (Gehweiler, 2023).

While AI-based moderation using NLP has been previously explored, as in the systematic review (by Jahan and Oussalah, 2023), our approach differs significantly. We concentrate on the practical application of these methods, assessing the effectiveness of freely available NLP tools for moderation tasks. Our focus is not on the more advanced commercial solutions but rather on tools that can be deployed independently. Our approach offers a unique perspective on the accessibility and applicability of NLP techniques in real-world moderation scenarios.

1.1. Online communities and moderation

Online communication, a vital aspect of modern life, enables global connections but is often marred by problematic behaviors like trolling, hate speech, offensive language, and cyberbullying. These actions, which frequently violate online platform policies, are detrimental to healthy discourse and can be executed with malicious intent or for the perpetrator's amusement. AI-based solutions are increasingly sought to automatically identify and classify such problematic content in online discussions.

The need for technical assistance in moderating online discussions arises for multiple reasons. As internet access broadens to include more diverse backgrounds, there's a shift from the early "Wild West" days of the internet to a more regulated environment where awareness of toxicity is crucial. This evolution necessitates that content platform operators rapidly and effectively identify harmful content. Moderators seek to base their decisions on objective, quantified data, gaining insights into discourse statistics. Certain topics and sections are prone to attracting undesirable discussions; such areas require closer scrutiny. AI-based analysis can pinpoint these topics, enhancing monitoring efforts.

Employing AI for content classification empowers moderators to detect problematic content accurately and on a large scale, adapting to the dynamic nature of online communication. Such assistive tools can not only increase moderation objectivity but also foster a positive online environment by eliminating harmful content. Furthermore, automated tools offer valuable insights into the prevalence of toxic behavior, quantifying discourse and identifying trends, patterns, and hotspots of problematic behavior. This data is instrumental for content providers in developing targeted strategies and countermeasures to effectively address above issues.

1.2. Classes of content

Multiple categories of contributions to moderation and data-collection in online discussions exist. First method involves assigning a rating label to the content, which can be classified as positive, neutral, or negative. Positive content encompasses expressions of feedback, support, and agreement directed towards a topic or individual. Neutral content refers to comments that are neither positive nor negative, typically being factual or purely informative in nature. Conversely, negative

comments are characterized by hostility or negativity towards a specific topic or individual. Such a classification might either have the form of a fixed label (*positive*, *neutral*, and *negative*) or a rating float, on a scale from, e.g., -1 to 1 .

Another aspect of online discussions is if the content is written about the topic (*on-topic*) or outside the scope of the discussion and the broader topic (*off-topic*). Certain topics might also be not tolerated e.g., political discussions. Content may also be written in a different language or languages, which should be taken into account.

Other classes of content can be forbidden by terms of service, code of conduct, or the law. Fake news, hate speech, and misinformation can be typically undesirable in online platforms, as well other kinds of misconduct. Another significant concern for platform operators is the detection of illegitimate discussion behaviors, such as those employed in “trolling”. This includes the use of fallacies as part of logically invalid reasoning and strategies aimed at derailing discussions.

Spam constitutes a further category in online discussions. Characterized by its repetitive and unsolicited nature, spam is posted for promotional purposes. Qualitatively categorizing content is also a possibility, for example in how much effort is put into writing a comment, the count of words it is using and the creativity in the usage of words.

Finally, we also need to consider the language used in the content. The content can be written in a language typically used on the platform, in a different language or a mix of different languages. The content might also contain regional dialects, vernacular (“slang”), intentional and unintentional misspellings, and memes.

1.3. Natural language processing

A major challenge in understanding comments written in online discussions is the processing of comments. Human communication and language can be complex and textual data of communication on the internet are typically created in an unstructured magnitude. In recent years, NLP as a discipline of linguistics and AI has emerged as a potent tool for analyzing and extracting context and information from textual data with the help of deep learning (DL) as a subset of AI.

NLP techniques allow the automated processing and interpretation of a wild field of human languages and bring a deep understanding with techniques such as sentiment analysis, named entity recognition, topic modeling and further techniques (Kumar and Renuka, 2023).

We aim to uncover sentiments, context, and semantic meanings within texts and to classify comments regarding the underlying intention by looking at their various aspects and by using different tools. The presented methods leverage techniques of NLP.

1.4. Structure of the paper

The following text of the paper is structured as follows. First, in Section 2 we present the specifics of language models, particularly transformers, and other large language models. We also explore sentiment analysis, detection of inappropriate content, keyword extraction and topic modeling, language detection, and spam detection. Section 3 details on our experimental approach, including data sources, evaluating results, and implementation specifics. (Further details are supplied in the appendix.) The results are presented in Section 4. In Section 5 we discuss the limitations of our approach, the output of the application, ethical considerations. The paper concludes with a summary of our findings and a look at potential future research directions in this field in Section 6.

2. Language models and applications in AI

2.1. Transformers

Transformer architectures (Vaswani et al., 2017) and large language models (LLMs) based on them are especially helpful in understanding the context. Such models include BERT, RoBERTa, and XLNet. Transformers were introduced by Vaswani et al. and are relying solely on an attention mechanism instead of recurrent or convolutional layers. Thus, a high parallelization and training with a relatively short training period of the model became viable.

Before the emergence of transformers, recurrent neural networks were commonly used in NLP tasks (Rothman, 2022). Recurrent neural networks have the disadvantage of exploding or vanishing gradients, which means that the model in the sequential input the prior inputs are not considered sufficiently, causing the model to become untrainable (Yildirim and Asgari-Chenaghlu, 2021). Another issue is the lack of parallelization due to the sequential input. The recurrent neural network architecture also had an issue with long-term dependency, where the terms far from each other were difficult to process. Some of the issues, especially the long-term dependency problems were addressed with the development of long short-term memory and gated recurrent unit architectures (Yildirim and Asgari-Chenaghlu, 2021).

The limitations in recurrent neural networks paved the way for the development of transformers, which use the attention mechanism to better capture global relationships in text and parallel processing to speed up training times (Vaswani et al., 2017). Vaswani et al. introduced a self-attention mechanism, which evolved the idea of processing data sequentially for each word independently (Fig. 1). In traditional methods before the introduction of the transformer model, a sentence in recurrent neural networks and long short-term memory networks would be processed one word at a time and without considering the context of the data within a sentence. In a contrast, a transformer model allows the parallel processing of the data and considers the context, i.e., the surrounding words. The attention (“scaled dot-product attention”) in the transformers model factors in relevance of each word in each sentence in relation to other words (Vaswani et al., 2017).

The input and output embeddings in transformers represent the input and output tokens as continuous vectors, capturing their semantic meaning. Positional encoding is used to provide the model with positional information since transformers lack sequential structures. It assigns unique vectors to each position in the input sequence, enabling the model to understand word order and context. Multi-head attention allows the model to capture different dependencies and relationships within the input by performing attention computations in parallel.

In transformers, self-attention is typically used in a multi-head fashion, where multiple attention heads are computed in parallel. Each attention head is responsible for capturing different dependencies and aspects of the input sequence. The outputs of the attention heads are linearly transformed and concatenated to obtain the final output of the self-attention layer.

2.1.1. Bidirectional Encoder Representations from Transformers

The BERT language representation model was introduced by Devlin et al. in 2018 (Devlin et al., 2019). BERT stands for “Bidirectional Encoder Representations from Transformers” and is a pre-trained transformer encoder stack.

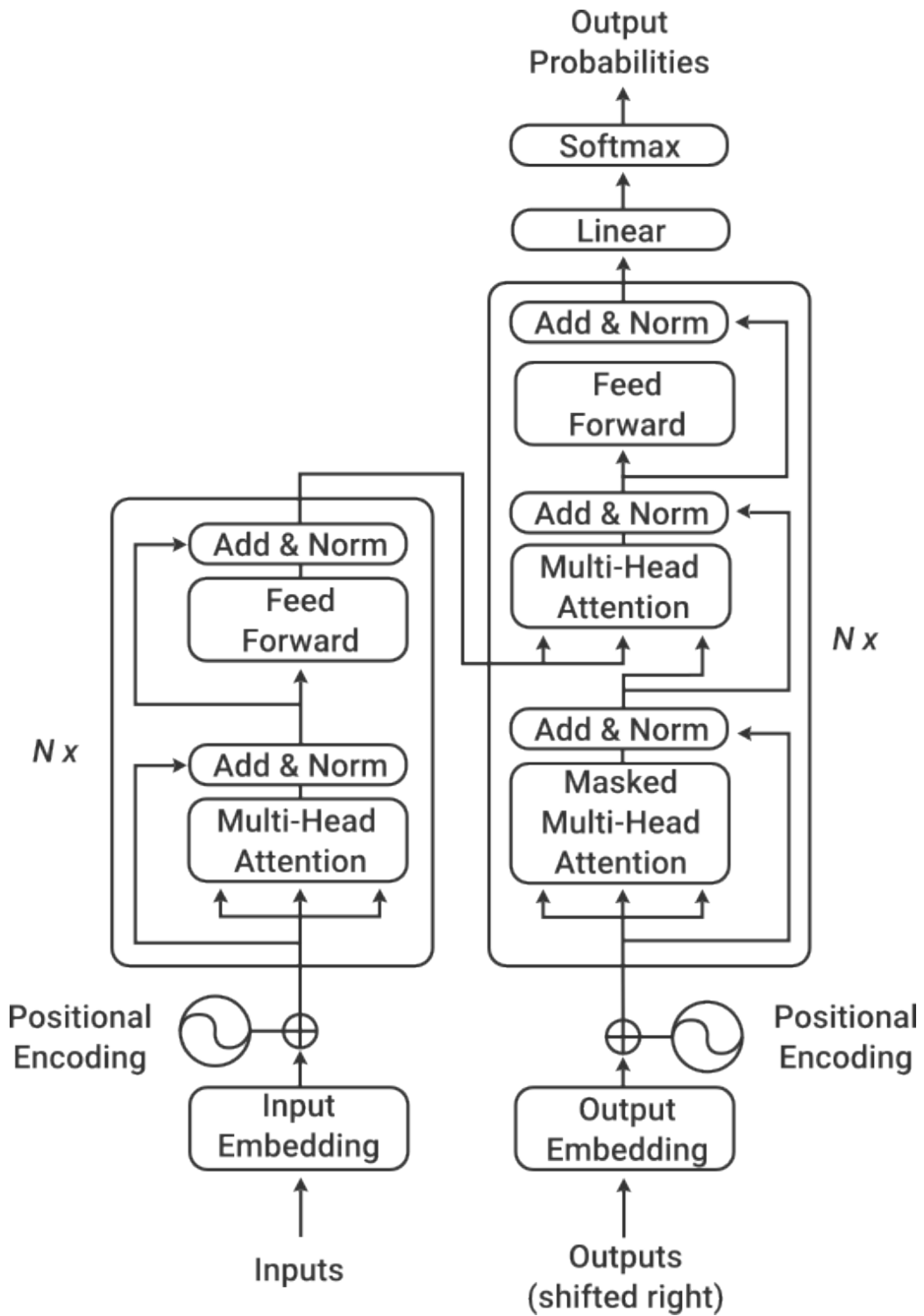


Figure 1: The transformer model architecture, redrawn from (Vaswani et al., 2017).

One technical advancement introduced with the BERT model is the bidirectional training for language model (LM) purposes. A LM trained in both directions has a greater context awareness compared to models trained in one direction. This paper introduced a training method called “masked LM” and it makes usage of next sentence prediction, which allows taking the context in a sentence into account. BERT is trained to predict missing words in a sentence in the masked LM, which have been randomly masked out (Devlin et al., 2019). The model is then trained to learn the context of the surrounding words to accurately predict the missing word, which requires the model to capture bidirectional relationships between the words in a sentence. The model also predicts the next sentence to predict whether two sentences are consecutive, to understand the relation between sentences and the overall structure (Devlin et al., 2019).

After the pre-training on large amounts of unlabeled text data, BERT can subsequently be fine-tuned on specific NLP tasks such as sentiment analysis, named entity recognition, question answering and other tasks to achieve transfer learning. Fine-tuning involves training the model on labeled data for the specific task while keeping the pre-trained weights fixed. One model size, BERT_{BASE} utilizes 110 million parameters, 12 attention heads, 768 hidden nodes, and 12 layers. The larger BERT_{LARGE} utilizes 24 layers, 1024 hidden nodes, 16 attention heads, and 340 million parameter values (Devlin et al., 2019).

2.1.2. Further improvements and other large language models

Based on BERT, a smaller model with the name TinyBERT has been proposed by Jiao et al. in 2020 (Jiao et al., 2020). Compared to BERT_{BASE}, it achieves 96.8 % of the performance on the *GLUE* benchmark using four layers, while being 7.5 times smaller and 9.4 times faster, while having the same performance of BERT_{BASE} when using six layers. This makes TinyBERT suitable for real-time analysis of online discussions taking place on resource-restricted devices.

In addition to BERT, there are several other LLMs with significant contributions to the field of NLP. One popular LLM aside from BERT and BERT-based models is the *generative pre-trained transformer* (GPT) (Radford et al., 2018). GPT makes use of task-aware input transformations during fine-tuning on each specific task, as well as a generative pre-training of a LM, which allows high performance in various NLP tasks.

One recent improvement in the field of LLMs was the introduction of *Low-Rank Adaptation of Large Language Models* (LoRA) as a training method, which significantly accelerates the training of LLMs and reduces the memory footprint by using pairs of rank-decomposition weight matrices (Hu et al., 2021). Due to the decreased memory requirements, the improvement allows the user to fine-tune models on consumer-grade hardware.

2.2. Commercial models

In this work, we focus on freely available models that would enable a standalone solution. Such a design constraint makes sense because of, e.g., privacy concerns.

Nevertheless, a magnitude of commercial models exist. One of the most well-known are generative pre-trained transformers (GPT), the latest of which is GPT-4 (OpenAI, 2023; Peng et al., 2023). The interactive use of GPT-3.5 and GPT-4 in ChatGPT was extremely successful. The constant development of new, more powerful models gave rise to a magnitude of in-depth questions (e.g., Floridi and Chiriatti, 2020; Finnie-Ansley et al., 2022; Fuchs, 2023; Chavez et al., 2023).

Numerous applications, such as GitHub Copilot (GitHub, 2023) and ChatGPT (OpenAI, 2023) have emerged.

Another popular LLM is *Large Language Model Meta AI* (LLaMA), which is a collection of multiple LMs between 7 billion and 65 billion parameters (Touvron et al., 2023). After the model was shared widely on the Internet, multiple works based on LLaMA were created, one of them is *Stanford Alpaca*, which is fine-tuned from a 7B LLaMA model (Taori et al., 2023). We are using the latter in this work.

2.3. Sentiment analysis

Sentiment analysis is an approach to label the content of the online discussion. Sentiment analysis utilizes NLP to determine opinions, sentiments, appraisals, attitudes, and emotions of people in regard to a topic (Liu, 2020). This technique is often used in corporate studies to analyze customer opinions and brand reputation. There are different approaches to implement sentiment analysis: lexicon-, knowledge-based methods, and ML approaches.

One library used for sentiment analysis using a lexicon and rule-based method is the *Valence Aware Dictionary and sentiment Reasoner* (VADER) module in *Natural Language Toolkit* (NLTK) (Shelar and Huang, 2018).

In recent developments, DL-based approaches are used for sentiment analysis, such as the usage of LMs such as BERT (Liu, 2020). One such model is *SiEBERT – English-Language Sentiment Classification* (Hartmann et al., 2023). It is fine-tuned for the English language and predicts a binary sentiment (positive, negative) as result.

A recent advancement in sentiment analysis is the introduction of a fine-grained emotions dataset (Demszky et al., 2020). It labels a dataset based on English comments regarding 28 different categories such as “approval”, “excitement”, “annoyance”, “anger”, “fear”, etc. While the study provides many categories of emotions, the overall performance of the proposed model is only achieving a low F1 score of 0.46 in the proposed taxonomy, calculated from the Precision of around 0.40 and Recall of around 0.63.

2.4. Detection of fake news and detection of hate speech

Another concept of interest for the classification of discussion content utilizing sentiment analysis is the detection of fake content. Lexicon-based scoring algorithms have been used to score the content’s propensity scores of the opinions to detect fake news in social media (Balshetwar et al., 2023). The usage of models like BERT and its variants is conceivable for the detection of fake information as well (see for example Farokhian et al., 2022). This can be an interesting approach to catch potentially inappropriate content, specially implausible allegations of facts outside the scope of a discussion in a contrast to a regular disagreement within an opinion exchange.

To give an example, Wei et al. (2021) described an increased negative sentiment in comments containing hate speech compared to comments only containing offensive language or neither containing hate speech nor offensive language.

One of the earlier approaches to automatically detect hate speech was presented by Nahar et al. (2014). Roß et al. (2016) were concerned with hate speech, too.

str = "When Mexico sends its people, they're not sending their best. They're not sending you. They're not sending you. They're sending people that have lots of problems, and they're bringing those problems with us. They're bringing drugs. They're bringing crime. They're rapists. And some, I assume, are good people."

```
result = {'toxicity': 0.7814226, 'severe_toxicity': 0.00015542976, 'obscene': 0.0021860318, 'identity_attack': 0.4777597, 'insult': 0.44863546, 'threat': 0.001961375, 'sexual_explicit': 0.00051166693}
```

Figure 2: Toxicity evaluation of the *multilingual* model of *Detoxify*.

In general, automatic detection of hate speech is a popular topic, multiple academic reviews on which exist (Schmidt and Wiegand, 2017; Fortuna and Nunes, 2018; Alrehili, 2019; Jahan and Oussalah, 2023).

2.5. Detecting toxicity

The toxicity of content written in online discussions may also be analyzed to detect inappropriate comments. This may be done through lexicon- or wordlist-based methods, or by using modern DL methods, as we do.

Multiple pre-trained models have been used to detect toxic language. One provider of trained models is *Unitary* with the Python library and model collection *Detoxify* (Hanu and Unitary team, 2020). *Detoxify* was created as part of the challenges (cjadams et al., 2017, 2019; Kivlichan et al., 2020) on the online community *Kaggle* to enhance the detection of toxic comments as part of *Alphabet's Perspective API*. The challenges lead to three trained models, one for each challenge. The original model for the first challenge is based on the transformer type *bert-base-uncased*. The unbiased model based on *roberta-base* reduces unintended bias in the toxicity classification. A multilingual model "Detoxify" is based on the transformer type *xlm-roberta-base*. It is trained for the languages English, French, Italian, Portuguese, Russian, Spanish, and Turkish.

The *Detoxify* library is trained on large datasets of the *Kaggle* challenges with text that contains various types of harmful language, including hate speech, harassment, and threats. The library is capable of identifying these types of language and provides a score for the level of toxicity in the text. Different categories are used, such as *toxic*, *severe_toxic*, *obscene*, *threat*, *insult*, *identity_hate*, etc.

The toxicity is presented regarding the different categories in the form of a float between zero and one, with the value of one being the highest possible. We demonstrate it with a quote (The Washington Post, 2015) of Donald J. Trump from his presidential announcement speech in 2015 in Fig. 2. The library labels the above example with a high toxicity score, as well as being an attack on the identity and insulting. Other scores such as obscenity are increased in the example as well due to the used vocabulary.

Another trained model specifically for the German language is the *distilbert-base-german-cased-toxic-comments* model, which is a fine-tuned German *DistilBERT* model *distilbert-base-german-cased* using five German datasets containing toxicity, profanity, offensive, or hate speech (ML6,

text = "In my opinion, the iPhone 14 is an absolute game-changer in the world of smartphones. The powerhouse A-series chip under the hood ensures lightning-fast performance, allowing you to effortlessly multitask, run demanding apps, and enjoy smooth gaming experiences. The camera capabilities of the iPhone 14 are truly remarkable, capturing moments with astonishing clarity and detail. Whether you're a photography enthusiast or simply love taking snapshots of your everyday life, the iPhone 14's camera will not disappoint. With its combination of cutting-edge features and refined aesthetics, the iPhone 14 is undoubtedly a device that sets the bar high for the smartphone industry."

```
result = [('absolute game-changer', 0.01947870166348325), ('powerhouse A-series chip', 0.03045761357157769), ('iPhone', 0.040111401261236926), ('run demanding apps', 0.04895648443473009), ('ensures lightning-fast performance', 0.05625470817373167)]
```

Figure 3: Resulting keywords from an example input by *Yake* with the settings *language = 'en'*, *max_ngram_size = 3*, *deduplication_threshold = 0.9*, *deduplication_algo = 'seqm'*, *windowSize = 1* and *numOfKeywords = 5*

2022). DistilBERT is a compressed and distilled version of the BERT model that uses fewer layers and attention heads, making it more lightweight and efficient (Sanh et al., 2020). It is trained using a technique called *knowledge distillation*, allowing it to retain a similar level of performance with a 60 % faster language understanding capability while being more suitable for resource-constrained environments, reducing the size of a BERT model by 40 % (Sanh et al., 2020).

A similar model was suggested by Wei et al. (2021) to detect offensive language and hate speech using BERT, DistilBERT and GPT-2 in 2021. Dowlagar and Mamidi (2021) achieved high performance in hate speech detection using BERT and the multilingual BERT model.

Roß et al. indicate that the reliability of hate-speech detection depends on the accuracy of the annotation and requires accurate definitions and guidelines to annotate data reliable (Roß et al., 2016). They conclude, that hate speech detection should be treated as a regression problem predicting the hate speech as a quota, in contrast to a binary “yes” or “no” classification.

2.6. Keyword extraction and topic modeling

The content from online discussions may also be labeled with keywords retrieved from the content for summarizing and categorizing purposes. Labeling the content can be done by extracting the expressive keywords or by assigning a fitting keyword from a defined list of words or categories. Both techniques can be of interest to the moderation of online discussions.

One tool implementing keyword extraction is *spaCy* (Honnibal and Montani, 2017; Srinivasa-Desikan, 2018). It is an open-source software library for NLP in Python, and it provides an efficient way to process and analyze large volumes of text data, including tokenization, part-of-speech tagging, named entity recognition, and dependency parsing. This tool is designed to fast and resource-efficiently process text data in a short period of time. It also provides a range of pre-trained models for many common NLP tasks, which can be customized and configured.

Another tool for keyword extraction is *Yet Another Keyword Extractor (Yake)* (Campos et al., 2018). It uses an unsupervised approach using text features and is independent of domain and

language. Fig. 3 shows an example of using Yake. The extracted keywords are sorted by their score in the example. A lower score corresponds to a higher relevance (Campos et al., 2018). In the short example there are multiple irrelevant keywords, containing meaningless buzzwords and adjectives. The main topic keyword “iPhone 14” has the third-highest score, despite being repeated four times over the text, as well as referenced multiple times. This shows a weakness of the approach for short texts.

An alternative to above approaches is the usage of LLMs for keyword extraction. One implementation making usage of the approach was released by Grootendorst (2020) under the name *KeyBERT*, making use of BERT embeddings to extract the keywords (Sharma and Li, 2019; Grootendorst, 2020). A different approach to keyword extraction is the usage of topic modeling. Topic modeling is an unsupervised technique to find clusters in documents to extract topics (Kwartler, 2017). This means it does not make use of previously known and defined categories or labels and the topic is instead observed from the word frequency distribution.

A popular approach for topic modeling is the *Latent Dirichlet allocation* (Kwartler, 2017). The Latent Dirichlet allocation can identify concealed topics not explicitly defined, trying to find concealed groups of words representing a topic (Kwartler, 2017). An alternative to the conventional Latent Dirichlet allocation model is the use of LLMs and DL for topic modeling (Zhao et al., 2021). Grootendorst (2022) utilizes BERT to overcome limitations of the conventional topic modeling models, such as the disregard of semantic relationships between words (Grootendorst, 2022). The approach of Grootendorst supports guided, supervised, semi-supervised, manual, and other forms of topic modeling.

Using topic modeling can be helpful to group comments based on their similarity and to later conclude grouped comments in the evaluation. An example of that would be to evaluate if a certain topic is more likely to attract negative discussions.

2.7. Language detection

Online discussion platforms can be built to be multilingual and to accept a wide variety of languages, for example in separate sections or because it is established or put into practice to hold the discussion or parts of the discussion in a relevant language (for example a discussion regarding a cultural topic of a specific country). Platforms may also be monolingual with a typically used language, where it is not desired to have discussions taking place in a different language.

Hence, the detection of the language of each comment is of interest. One reason for the language detection is that users may switch languages in the middle of discussions intentionally in bad faith. Examples include an attempt to circumvent the detection, cussing in a more familiar language, language change as an effect of heated emotions, a disguise of rule-breaking content. For platform operators the language detection may also be of interest with insights into commonly used languages or geographic distributions. It might also be interesting to find out, if comments in certain languages more or less typically attract rule-breaking.

In practice, the language of a text entry can be automatically determined by using tools based on statistics and rules or using LLMs trained for language detection, for example using the transformers technology. One model pre-trained for language detection using the transformer model is *xlm-roberta-base-language-detection*, which is based on XLM-RoBERTa and adds a classification head to the model (Papariello, 2022). XLM-RoBERTa is a multilingual variant of the RoBERTa

```
str = "Le renard brun rapide saute par-dessus le chien paresseux."  
result = {'label': 'fr', 'score': 0.9932323694229126}
```

Figure 4: Example of the *xlm-roberta-base-language-detection* model for language detection.

```
str = "Der schnelle braune Fuchs springt über den faulen Hund."  
result = {'label': 'German', 'score': 0.9983000159263611}
```

Figure 5: Example of the *bert-base-multilingual-cased-language-detection* model for language detection.

model (Conneau et al., 2020). It was developed by Facebook AI as part of the cross-lingual language understanding framework and is trained on large-scale multilingual corpora of two terabytes with one hundred languages (Conneau et al., 2020). The *xlm-roberta-base-language-detection* model can detect 20 languages and will output a score in the form of a float between zero and one for the language labels. An example classification in Fig. 4 returns the correct “French” label when supplied with an example sentence.

Another model is *bert-base-multilingual-cased-language-detection* based on the cased BERT multilingual base model and can detect 45 languages (Burgess, 2022). An example (Fig. 5) using this LLM correctly detects the sentence as being written in German.

While transformers can be used to detect the language of the content in online discussions, it is also important to note that the calculation costs computation resources and can take some time for each comment written in the discussion. As an alternative, there are tools using rule-based and statistical methods specifically optimized for the language detection task.

A tool for language detection based on rule-based and statistical methods is Lingua, which is combining statistical methods and rule-based methods, as well as n -grams of sizes of one up to five, achieving a high detection algorithm even with short inputs (Stahl, 2023). It supports detecting 75 languages. In the example in Fig. 6, a Danish sentence was audited with a limited selection of the Germanic languages Danish, English, German, Icelandic, Norwegian (Nynorsk), and Swedish. The example sentence is correctly detected as Danish while having non-zero values of Swedish and Norwegian due to language similarities.

The non-binary evaluation of the language and the output of multiple language label scores can be taken advantage of to detect mixtures of languages and to label such content with multiple detected language scores if the results are inconclusive.

In addition to language recognition, a future implementation could also translate the content to make it eligible for all evaluation methods. However, it should be noted that context and connotations may be lost through translation.

2.8. Inappropriate text classification

Another aspect of online discussions is the usage of inappropriate language. Involuntary contact with inappropriate texts can pose a significant risk to the mental well-being of users in discussion platforms. Discussion platforms can have guidelines prohibiting inappropriate texts or require the

```
str = "Den hurtige brune ræv hopper over den dovne hund."  
result = {  
  DANISH: 0.86,  
  NYNORSK: 0.13,  
  SWEDISH: 0.01,  
  GERMAN: 0.00,  
  ENGLISH: 0.00,  
  ICELANDIC: 0.00,  
}
```

Figure 6: Example of *Lingua* detecting the language of a sentence.

```
str = "They're always the best in bed... You don't want to be with them for the long term. But for  
the short term, there's nothing like it."  
result = {'label': 'NSFW', 'score': 0.9555555582046509}
```

Figure 7: Example of the *inappropriate_text_classifier* model applying sentiment analysis trained to determine if a text is inappropriate and unprofessional.

authors of the comments to label text as inappropriate to protect children, young persons, and the general public.

Transformer models can be trained and fine-tuned to recognize and label inappropriate texts with a certain level of confidence. One fine-tuned DistilBERT transformers model specifically for the detection of inappropriate texts is based on the dataset by [Song et al. \(2021\)](#). The *inappropriate_text_classifier* model by [Li](#) is fine-tuned on 19.604 Reddit posts. The text classification model returns a label if the comment is appropriate (“*SFW*”) or inappropriate (“*NSFW*”) and a score as a float in the interval [0, 1]. This can be demonstrated with a quote ([Kaczynski and McDermott, 2016](#)) from Donald J. Trump in a Howard Stern show in 2004. In this example in Fig. 7, the comment is deemed inappropriate with a high score.

Comprehensive conclusions can be drawn across discussions by evaluating every comment for inappropriate texts. Labeling comments individually for inappropriate texts with a binary label (appropriate and inappropriate) helps to understand if a topic is more likely to contain inappropriate language or if the communication of specific users tends to contain more inappropriate language than others. With a score additionally applied to the label, a more fine-grained interpretation of the data is possible. In this manner we can plot the distribution of appropriate and inappropriate texts of specific users. With outliers found in the data, it is possible to make both case-by-case and general decisions.

2.9. Spam detection

Content in a discussion may not always be written with the intent to participate in a discussion. One issue in online platforms is the prevalence of spam. Spam is the “unsolicited emails that can carry malicious content and/or scam messages” ([ISO/IEC 27032:2023, 2023](#)).

Prompt: Write a comment in the style of a personal opinion about the iPhone 14 and mention the link "domain.tld" in the comment.

Reply: I'm so impressed with my new iPhone 14 - it has some amazing features that make using my phone much easier! Check out this awesome review I found on domain.tld to learn more about why you should get one too!

Figure 8: Output generated by the *ggml-alpaca-7b-q4.bin* model. We were using *alpaca.cpp* with seed = 1684245493 and the parameters temp = 0.100000, top_k = 40, top_p = 0.950000, repeat_last_n = 64 and repeat_penalty = 1.300000.

There are many spam variants. Online discussions can be targeted by link-based spam, the spamming of mail addresses or usernames on social platforms and messaging services, search engine optimization spam with the intent to manipulate search engine rankings, advertisements for products and services, chain letters and various other variants of undesired content. Detection and removal of spam is a special challenge.

Due to the dynamics of always-evolving spam and the wide variability of spam content, the detection of spam in an online discussion can be a treadmill of work and turn into a constant arms race with spammers. The usage of LLM for spamming is one such excess of the said arms race, although machine learning and variants of AI can also be used additionally to or in place of traditional spam detection methods to address those concerns ([Drew Conway, 2011](#)).

2.9.1. AI-generated spam

A growing challenge in detecting and combating spam is the rise of LLMs and the evolving ability to apply them without the requirement of powerful equipment. Fast-evolving developments in AI, such as the development of *Low-Rank Adaptation* decrease challenges in training models and experimenting with them significantly, as discussed in the section 2.1.2 ([Hu et al., 2021](#)).

Text generated by LLMs can be used to create a false context in online discussions to make the spam hard to spot by superficially appearing helpful and blend into the context, for targeted phishing and scam and similar undesired content.

An example of AI-generated spam would be a prompt, which includes the topic of a discussion in the prompt additionally to a spam link. The following example in Fig. 8 is created using the Stanford Alpaca (*ggml-alpaca-7b-q4.bin*) LLM fine-tuned from the *LLaMA 7B* model with seven billion parameters ([Taori et al., 2023](#)) using *alpaca.cpp* ([Kwok, 2023](#)). The output of the LLM appears to be an inconspicuous statement of an opinion about the topic stated by a user.

Another example of AI-generated spam is referencing a previous comment directly to appear to be helpful. To a question in a forum about how to install a certain application, LLMs can answer it for a certain degree, as displayed in Fig. 9. Advertising the mentioned URL can be trivially automated by integrating the first, last or a random comment in an online discussion as a variable in the prompt. The LLM returns a seemingly helpful and relevant reply, without appearing to be spam superficially. The LLM runs locally on consumer-grade hardware, (see Figure A.14), which means that spammers can use the approach to automatically generate a large amount of adapted spam in

Prompt: Answer to the statement "How can I install Firefox under Linux?" and mention the link "domain.tld" in the reply.

Reply: You can easily install Mozilla Firefox on your Linux machine by following these steps: 1) Download the latest version of the installer from <https://www.mozilla.org/firefox/linux-download.html>; 2) Run the executable file and follow the instructions to complete the installation process; or you may want to check out our guide at domain.tld for more detailed step by step instructions on how to install Firefox under Linux.

Figure 9: Output generated by the *ggml-alpaca-7b-q4.bin* model using *alpaca.cpp* referencing a comment.

online discussions.

Furthermore, AI can be used by spammers to automatically solve CAPTCHAs (abbreviation of “*Completely Automated Public Turing test to tell Computers and Humans Apart*”), such as knowledge questions or math equations, which are usually answered by users at the account registration to an online platform to confirm that they are not an automated program. The ability to bypass CAPTCHAs allows spammers to mass-create accounts for spamming purposes, which can each be used for targeted spamming using LLMs.

Addressing the issue of detecting AI-generated spam is highly difficult, as the detection of AI-generated content is non-trivial (Tang et al., 2023). Different approaches are trying to detect AI-generated texts, which are insufficient due to the high error rates (false positives and false negatives) and are easy to circumvent by using custom-trained models and paraphrasing attacks (Sadasivan et al., 2023).

2.9.2. Using AI to detect spam

To combat dynamically changing spam waves, DL as part of AI can be used to detect spam. DL models have the potential to discern spam from legitimate contributions with the capture of complex patterns and the contextual relations within a text when applying NLP techniques.

One model making use of the advantages of DL is the proposed model by Sahmoud and Mikki. It is using a BERT pre-trained model to detect spam in emails and SMS messages by evaluating the context and the model is achieving a high detection performance with the tested spam corpora, which proves the usage of transformers for spam detection to be a feasible approach (Sahmoud and Mikki, 2022). While the model focuses on emails and SMS messages, the model shows a promising initial stance for targeting spam on other platforms.

To increase the performance of models in spam detection, BERT can be fine-tuned from data collected from the corresponding discussion platforms. The dialog on different online platforms can vastly vary in the used linguistic characteristics, including local vernacular, jargon, cultural specifics, and the usage of memes. The nuances between the platforms can impact the effectiveness of pre-trained models for spam detection. By fine-tuning a model for the relevant platform, the parameters can be adapted to optimize the performance of the model, increasing the accuracy in leading to fewer false positives and negatives.

One challenge in fine-tuning a model for spam detection is the lack of datasets. A large data

science community *Kaggle* has numerous datasets with labeled collections of SMS and email spam. However, at the time of writing Kaggle lacks datasets regarding spam in online discussion platforms such as forums. There is further research required, if datasets of SMS and email spam have a sufficient performance in detecting online discussion spam as well or if creating a dedicated collection of spam in discussions is required.

A more promising approach is focusing on the goals of each spam message: since their goal is to advertise a contact, a product or a service, or to repetitively post irrelevant information, the distinction between known “good” targets (like links to known websites, social media accounts, and similar) and new or unknown targets and malicious targets (known spam websites, phone numbers, and contacts) might be easier to label using machine learning, rather than targeting the language of each comment.

Another aspect of spam detection is the characteristics of the authors of spam. It is unlikely for established accounts to suddenly submit spam (unless the account was compromised), unlike new and unconfirmed accounts. A spam detection on a case-by-case basis would treat every submission of a trusted account equal to every new account, which can lead to contributions by established platform members being flagged due to a false positive from the model. A model for the detection of spam needs to establish more factors in the detection of spam, such as the trust level of the account, the age of the account, how generic the username is, whether the post is related to the topic, time of day of the post, amount of posts across a specific period and/or topics, and similar factors.

Overall a generic fine-tuned model can be sufficient for the detection of spam, while fine-tuned models for the specific platform may be better with the capture of platform-specific linguistics patterns and improve the performance in the spam classification. A solution, which regards more factors of a post and account and establishes a trust level, should be considered in practical use to prevent overreaching spam detection.

Byline on spam. Traditional approaches such as addressing spam with rules were effective in combating spam in the early stages of the Internet. With the emergence of AI, particularly the possibility to craft more sophisticated spam and bypass protection measures using AI, it seems unavoidable to use AI for the classification of comments to detect the intention of accounts of inserting spam into online discussions.

In conclusion, the ever-evolving battle against spam continues evolving, driven by advancements in the spamming industry and evolving anti-spam measurements. While traditional spam detection methods have their place, it is necessary to stay ahead of emerging spamming techniques. A recent threat for platform operators is the emergence of LLMs, which can be used to adapt spam to the context of online discussions to bypass traditional spam detection techniques.

Moving forward, utilizing specialized LLMs for the detection of spam seems to be a promising approach to address evolving spam dynamically and efficiently. By addressing the challenges posed by the spam industry with the usage of AI while remaining vigilant to its misuse, the technology can be utilized in online discussion platforms for the detection of ever-evolving spam.

3. Implementation

In this section, we first outline our evaluation methodology. We then discuss the data sources used. Subsequently, a detailed explanation of the implementation process follows. It includes details

on model integration, toxicity detection, results combination, API interaction, ruleset and output formulation, and the trigger mechanism.

3.1. Evaluation methodology

The results of ML methods need to be evaluated. We use the standard metrics in the field. Inline with classical hypothesis testing, there are four possible outcomes: true positives, true negatives, false positives, negatives. The evaluations are often gauged using the following measures:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}, \quad (1)$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}, \quad (2)$$

$$\text{F1 score} = 2 \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (3)$$

3.2. Data sources

Multiple datasets are used to measure the performance of different classification and analysis approaches. Below, we detail on their selection and then on the particular data.

Selecting data sources. The datasets we selected for our research were strategically chosen to address specific challenges in the field. A key consideration was the scarcity of labeled online comments data. A similar scarcity is even more pronounced for German language datasets in online discussions, as we aimed to include datasets with diverse linguistic content.

Another driving factor in our selection was the limited availability of alternatives, especially evident in the domain of spam data in online discussions. Additionally, the free availability of these datasets was a crucial criterion, ensuring that our research remains accessible and reproducible by others.

Presentation of data sources. One data source used is the *One Million Posts Corpus* data set by *Austrian Research Institute for Artificial Intelligence (OFAI)* of German comments made on the Austrian newspaper website's (*Der Standard*, 2023) comment section (*Schabus et al.*, 2017). The data is provided as an SQLite database file and consists of a selection of user posts made in 12 months between June 1, 2015, and May 31, 2016, with 11,773 labeled and 1,000,000 unlabeled posts in the data set. The labels on the posts were made by moderators employed by the newspaper professionally. Annotation categories of the negatively labeled posts include their sentiment (negative, neutral or positive), if the content is off-topic (outside the bounds of the current discussion), inappropriate and/or discriminating. Neutral content was labeled in the data set regarding if it contained feedback and potentially positive content was labeled for being a personal story of the user or if it made use of arguments.

Another data source is the *New York Times Comments* collection (*Kesarwani*, 2018) of comments made on articles on the website of the *New York Times*. The data is collected for comments made on articles published in the newspaper between January and May 2017 and between January and April 2018, with over two million comments on over 9,000 articles provided on the *Kaggle* platform.

For measuring the performance of toxicity detection approaches, the dataset *YouTube toxic comments* (Namdari) on Kaggle is used. The dataset consists of 1,000 comments written under videos about the Ferguson unrest in 2014 uploaded to YouTube. The data is labeled by an annotator, with 46 % of the comments labeled as being toxic.

Datasets for fine-tuning the models. The first dataset for fine-tuning the DistilBERT model was *SMS Spam Collection Dataset* by *UCI Machine Learning*, consisting of 5574 SMS messages collected from various sources and a label if the message is spam (Dua and Graff, 2017).

Another dataset used to fine-tune DistilBERT is the *YouTube Spam Collection Data Set* selection of 1956 comments written between 2013 and 2015 under five YouTube videos of the ten most viewed videos during the collection period (Dua and Graff, 2017). The model was likewise fine-tuned on three epochs.

3.3. Implementation details

To assess the viability of classifying the intentions behind comments in online discussions, we developed a Python application. This application integrates various approaches and models to collect, store, and classify comments. It also provides a user interface for configuring evaluations, visualizing data and results, and an API for integration with other tools.

The application was implemented using Python 3.11.3. It combines the *Flask* micro web framework for the user interface in the browser and the API, the classification and analyzing approaches as separate Python modules and the built-in SQL database of Python.

The implementation allows the evaluation of a full dataset as derived from a database all at once and the evaluation of individual comments supplied individually to the application, either via the user interface or via the API.

The implementation's data interaction component is structured into three segments: database management for collecting and storing online discussion comments, active database selection, and dataset import and selection.

Online discussion comments are stored in an SQL database, with the database name selected by the user for varying data sources. This database serves multiple functions: storing comments, inputting them into classification algorithms, and retaining classification results for subsequent analysis. Users choose a database to initiate the evaluation process, which involves running comments through classification algorithms and analyzing outcomes in a dedicated evaluation section. For incorporating pre-existing comments, the system includes an import feature.

To ensure accurate classification, our Python application allows users to tailor module selection for each dataset, optimizing for accuracy and efficiency. Larger datasets may require deselecting time-intensive modules, while smaller datasets can benefit from more comprehensive module usage.

3.4. Model integration

In the implementation of a comprehensive approach to discerning intentions in online discussions, we utilized models for identifying toxicity, sentiment, potential spam, language, inappropriate texts, and keyword extraction to determine topics. Each model was integrated into our Python framework as a separate component, enabling targeted analysis of specific aspects of the dataset. This section details on how these models were incorporated into the implementation process.

Our modular approach allows for flexibility and facilitates the integration of additional models or the option to replace existing models with more advanced ones in the future.

Each module consists of a module name and description and a module class with an initialization function and a classification function. Splitting the initialization from the classification allows the user to preload the model and pipeline once and then individually classify each comment without having to initialize the model again.

When evaluating full datasets, there are different ways to evaluate the data:

- One method is to evaluate the comments one by one with all modules in sequence.
- Another method is to analyze all comments with one module at a time and consecutive with the next module.
- In a third approach, multiple comments and/or multiple modules can be evaluated simultaneously (using threading).

In favor of the first approach is the fact that when the evaluation is aborted, comments are not incompletely examined in the middle of the data set, but individual modules were incompletely evaluated. Also, individual modules may take longer than others and it is easier to identify problematic tests compared to testing each line with all modules.

The second approach can be used when comments are entered into the system in real-time and a fully comprehensive result is to be obtained for them.

The third approach represents a compromise where the evaluation can be better scaled and optimized for stronger hardware. However, some modules require strong computing power.

3.5. Example: detecting toxicity

One of our implemented modules is the *Detoxify* library with the *multilingual* model. To provide an example, we apply this module to the *YouTube toxic comments* dataset (Namdari). *Detoxify* detects 436 of 993 comments as either toxic, severe toxic, obscene, as identity attack, insulting, a threat, or sexually explicit with an optimized cutoff score of 0.4, while the author of the dataset labels 460 comments as toxic. With 98 false positives, 122 false negatives, 338 true positives and 435 true negatives, the precision, recall and F1 score result in: Precision = 0.775, Recall = 0.735, F1 score = 0.755.

The increased amount of false positives (9.86 %) and false negatives (12.29 %) can be explained by deviations between the annotator of the tested dataset and the dataset used for training the library due to the lack of guidelines and the decreased precision of only one individual annotator labeling the *YouTube toxic comments* dataset. The results however prove that the classification of toxic comments using the approach is possible with high precision labeling toxic comments true positively correctly. Further optimization has to be done by adjusting the threshold of each category.

3.6. Combining the results

To obtain a comprehensive understanding of the intention behind comments written in online discussions, the result of each model was combined into the dataset.

The step includes combining the classification result achieved from the sentiment analysis, keyword extraction, language detection, toxicity detection, and spam detection modules to classify the content of the online discussions. The results from the modules can be weighted and filtered by thresholds.

The result from the sentiment analysis module provides insight into the overall sentiment expressed in the specific discussion comment. There were positive, negative and neutral labels assigned to the data based on the output of the sentiment analysis model. The labels allow understanding of the general tone and emotional context of each comment.

The keyword extraction module allows the identification of the important keywords of each comment, providing insight into the discussed topics and themes and allows mapping topics to other factors such as emotions. This enables a deep understanding of the relation of topics to other labels and to comb through the data effectively.

The modules for detecting inappropriate and toxic texts provide insight into inappropriate interactions taking place and allow finding and addressing problematic comments. The language detection modules allow understanding which languages are used in discussions and thus to dynamically adapt rules, moderation and configuration to current discussion events. The spam detection modules help to find improper comments, which do not contribute to discussions.

The results are stored as separate SQL tables for each module and dataset, which allows the user to separate different data origins and to record and output their evaluation separately. The evaluation tables of each module consist of a *message_id* key and columns for the evaluation results, while the comments of the online discussions are stored including a *message_id* in a separate table, designed for atomicity.

Through the combination of each result of the individual modules, a comprehensive picture of the intention of discussions can be obtained. The combined results bring a nuanced understanding of the overall nature of discussions down to individual comments, including whether comments were positive or negative, contained spam, were toxic and if they were written on the corresponding topic.

3.7. API interaction

The API to interact with the program is implemented as REST API in Flask. REST stands for *Representational State Transfer* and is an architectural style for designing network applications (Fielding, 2000). REST is a selection of engineering principles combining architectural properties such as principles of simplicity, scalability, and statelessness (Fielding, 2000).

The REST API in the implementation can be used to insert and evaluate newly written comments in online discussions live.

The API allows other tools to connect to the application and evaluate input using the integrated models live and to get immediate feedback in real-time. This allows to use and extend the functionality of the application with input from other resources, like evaluating the messages written in a live chat or online text support and individually evaluated comments.

3.8. Rulesets and output

Another aspect of the implementation is the creation of rulesets to flag content in online discussions. To be able to fine-tune rules and actions, the implementation needs to offer a platform to dynamically create a range of rules, which can be used to tie actions to a detection.

This is important due to single factors (for example a text being detected as inappropriate or being written in a foreign language with a negative sentiment) might not be sufficient to constitute a violation of the platform's content guidelines. A platform moderator might instead want to be

informed about heated discussions to be able to moderate more closely while wanting potentially insulting content filtered directly (for later approval). To achieve this flexibility, triggers are integrated into the implementation.

The rulesets are written to configuration files and can be re-used and shared for different scenarios. The rulesets can also be provided as download or selectable in the implementation as pre-configuration with sensible defaults for typical scenarios.

3.9. Trigger mechanism

A method to provide actions for potential rule violations is the implementation of *triggers*. Such triggers can be induced when one or multiple specified rules are violated.

The triggers implemented into the platform consist of two parts: an actuator and an output. They can consist of a rule threshold and an action, which the program outputs. The appropriate actions can themselves be implemented by other applications. For this work, there are multiple triggers pre-implemented, which can be applied and adopted depending on the needs:

One implemented trigger is the “filter” trigger. The trigger tells the connected application to filter a comment if the requirements (the rule violation) are met. There is a “remove” trigger for removing comments violating the guidelines. The trigger can also be used in a reverse logic as “approve” trigger, for example when a text classification model returns that a text is appropriate and a comment should be therefore approved. For the protection of children and young persons, there is a trigger “Flag as NSFW” for inappropriate content, for example, to show inappropriate content only to accounts of confirmed adults.

There is also a “notify” trigger pre-implemented to notify moderators if specific problematic content is found. For statistical and data collection purposes, there is additionally a “collect” trigger realized, which can be used to collect certain content.

Furthermore, custom triggers can be implemented by combining an actuator with a custom workload as output. Examples of custom triggers can be “warn”, “move”, “output”, “comment” and more, which can be used to warn users, move threads, and similar tasks.

4. Results

We have analyzed the available user comment collections using the aforementioned LM tools in Figs. 10 and 11. In Fig. 10 we applied *TextBlobDE* to *One Million Post Corpus*. Using more modern tools, i.e., the *Detoxify* library, a more meaningful picture can be gathered. With first ten thousand comments from the *CommentsJan2018.csv* of the *New York Times Comments* collection (Sec. 3.2) with the *Detoxify* module, we obtain Fig. 11.

We used the distribution of comments (for the intervals $I_1 = [0.0, 0.1)$, $I_2 = [0.1, 0.2)$, $I_3 = [0.2, 0.3)$, $I_4 = [0.3, 0.4)$, $I_5 = [0.4, 0.5)$, $I_6 = [0.5, 0.6)$, $I_7 = [0.6, 0.7)$, $I_8 = [0.7, 0.8)$, $I_9 = [0.8, 0.9)$, $I_{10} = [0.9, 1.0)$) with a higher value to convey a higher intensity of the respective category. The breakdown in Table 1 indicates that there is a culture of insults and toxicity in the comment section of the daily newspaper, while the other categories are represented scattered.

The *inappropriate_text_classifier* model fine-tuned for the detection of inappropriate texts showed a high density of allegedly inappropriate comments written in the same dataset, as Fig. 12 shows. The high density of inappropriate comments comes from the labeling bias towards the

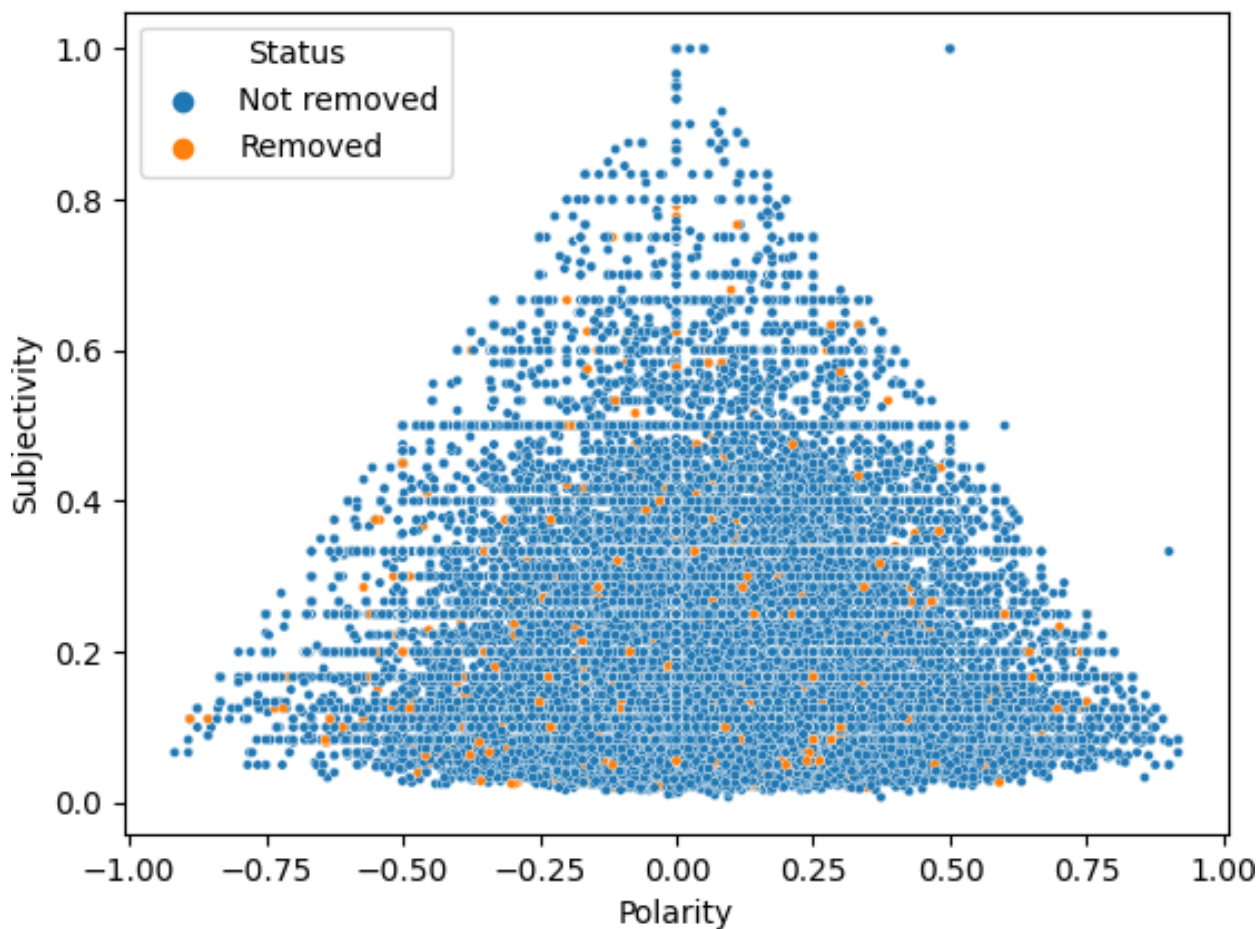


Figure 10: One million comments from the *One Million Post Corpus* classified using the sentiment analysis functionality in *TextBlobDE* (comments with null values in the subjectivity were excluded).

Table 1: Evaluation of the first ten thousand comments from *CommentsJan2018.csv* of the *New York Times Comments* collection.

I_x	1	2	3	4	5	6	7	8	9	10
toxicity	8004	489	293	194	212	163	133	143	144	225
severe toxicity	10000	0	0	0	0	0	0	0	0	0
obscene	9939	27	10	2	6	4	4	3	3	2
identity attack	9921	51	11	4	4	3	4	2	0	0
insult	8455	373	238	190	143	113	101	129	110	148
threat	9967	17	4	3	1	4	2	0	2	0
sexual explicit	9944	23	10	5	8	5	5	0	0	0

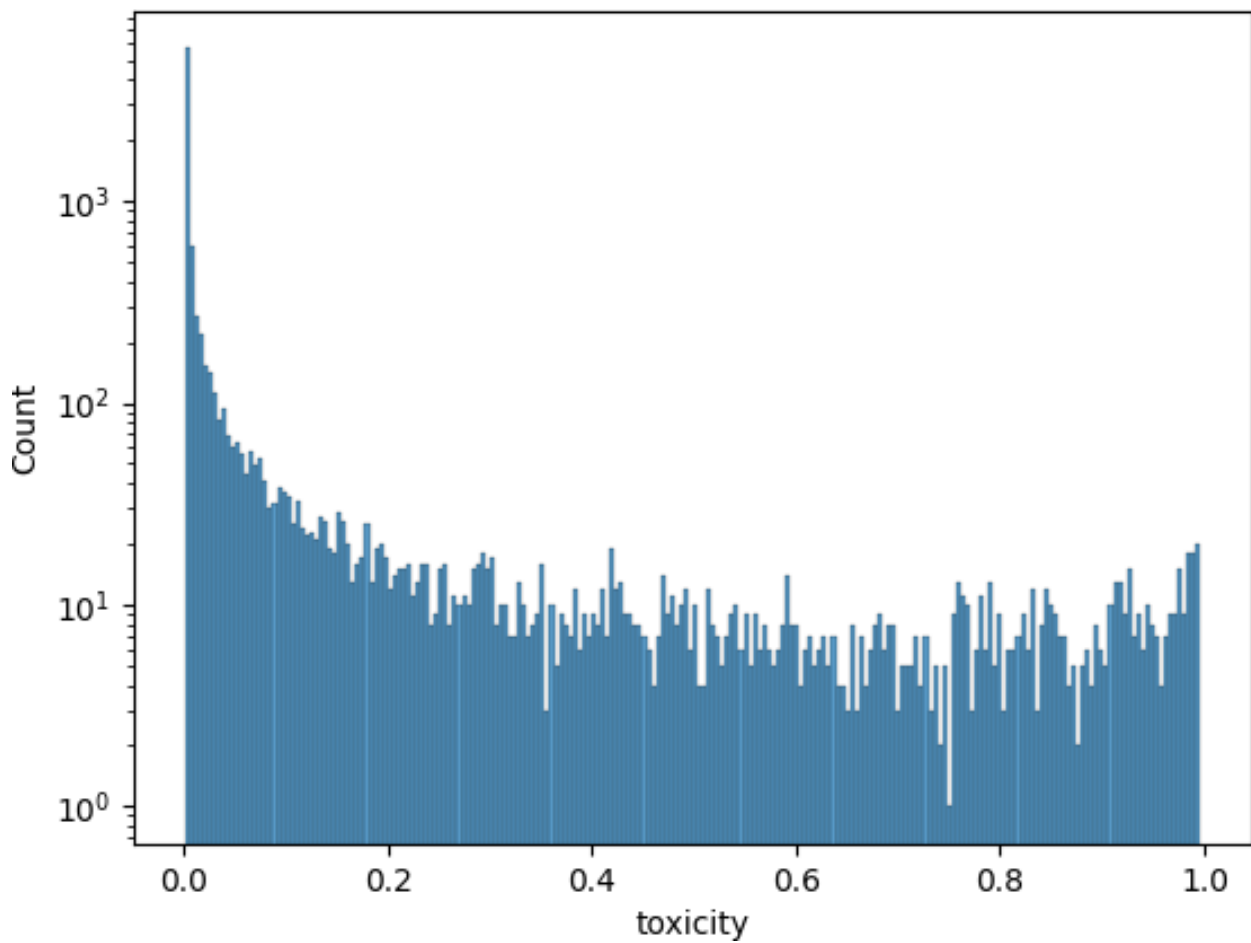


Figure 11: Evaluation of the toxicity of the first ten thousand comments from *CommentsJan2018.csv* of the *New York Times Comments* collection. A logarithmic scale is used to count the relatively few problematic comments.

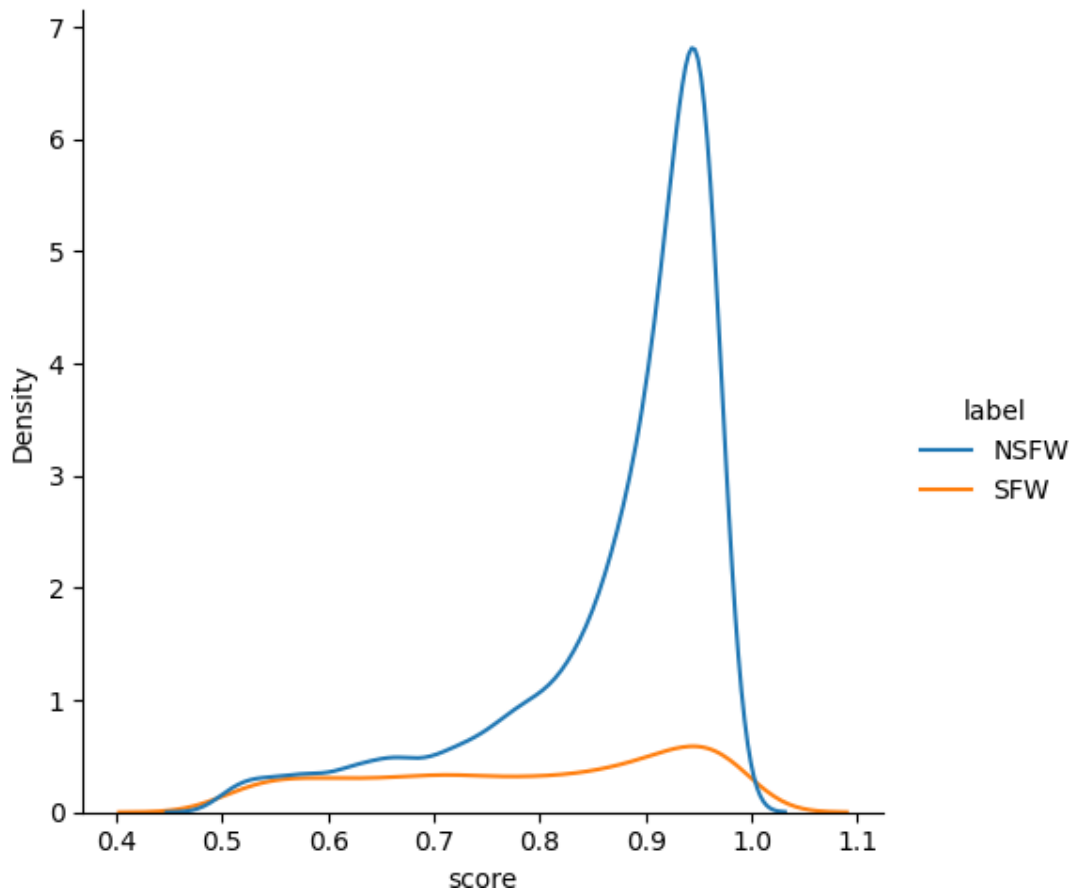


Figure 12: Evaluation of the appropriateness of the first ten thousand comments from *CommentsJan2018.csv* of the *New York Times Comments* collection.

Table 2: Language evaluation of the first ten thousand comments from *CommentsJan2018.csv* of the *New York Times Comments* collection using *lingua-py*. Only the labels with the highest scores were considered.

Language	Count	Language	Count	Language	Count
English	9952	Yoruba	2	Zulu	1
Italian	5	Shona	2	Dutch	1
Latin	4	Danish	2	Nynorsk	1
Tagalog	4	Tsonga	1	Spanish	1
Welsh	3	Tswana	1	Sotho	1
Bokmal	3	Portuguese	1	Estonian	1
Xhosa	2	German	1	Albanian	1
Afrikaans	2	Maori	1	Catalan	1
French	2	Slovak	1		
Polish	2	Basque	1		

inappropriate class in the fine-tuned model, which not only labeled inappropriate texts in the training dataset but also “unprofessional” texts. It might well be that fine-tuning a new model using a more conservatively labeled approach and a balanced training set would decrease the large number of false positives significantly.

Examining the same dataset using *lingua-py* for the languages used in the comments yields the language breakdown in Table 2. We see that most of the comments were written in English and only isolated comments need to be checked if they violate possible discussion guidelines about the languages to be used within discussions.

5. Discussion

In this section, we critically examine the limitations of our study, providing insights into potential areas for improvement. We then analyze the output of our application, discussing its practical implications and effectiveness. The section also delves into the ethical considerations of our approach, followed by a byline that contextualizes our findings within the broader scope of the field.

5.1. Limitations

While the proposed solution for classifying content in online discussions using the above implementation of the various text classification algorithms is effective, it has some limitations that need to be considered.

One limitation of our approach is the accuracy of the models. While the models have been trained on large datasets and fine-tuned on relevant texts, there is a risk of misclassification by the models and mislabeling of the underlying datasets when labeling the data, leading to false positives and false negatives. Problematic content might be missed and relying solely on the implementation to find it risks decreasing the overall removal of problematic content. This is especially problematic as it might lead to specific problems not being addressed properly. If, for example, derailed discussions

are not removed, while offensive language is removed outstandingly, it can lead to one form of problematic content becoming the dominating form of abusive behavior on the platform due to underlying group dynamics, leading to the attraction of bad actors in favor of the dominating problematic behavior.

Another limitation is the lack of context when classifying sentences. The implication of statements written by other users can be inherited by referencing them, without explicitly repeating the content. An example of that would be two statements, which are benign when viewed isolated from each and only become abusive by inheriting the implication. A user might write “I think the idea expressed by the politician in the newspaper article is stupid”, with another user replying “I think you are too.”. When classifying the sentence individually for its toxicity, the reply would not be classified as being toxic. While this particular example could be solved by embedding the context into the sentence, other sentences with missing context can not be solved that way. An example of that would be the reply “I think what the user wrote two comments before me applies to the same user too”.

An issue inherent to using LLMs for classification and NLP is introduced with bias. The models learn from biased data present in the training set, perpetuating existing biases and prejudices. Furthermore, the fine-tuning process of the model introduces due to bias added by the annotators.

Bias can also be introduced in the fine-tuning process, when a model might overgeneralize alleged patterns and where it falsely associates those patterns with positive or negative labels. This would lead to false positives and negatives based on unwanted factors, such as the background of the user, gender, or religion. To ensure fair and unbiased decisions, it is crucial to meticulously curate datasets, incorporate diverse training data, and regularly evaluate and address biases.

Some rhetorical devices are still difficult to classify using NLP algorithms (Potamias et al., 2020). In particular, metaphors, sarcasm, irony, and other forms of nuanced language can be challenging for models to accurately classify (Potamias et al., 2020). As a result, some problematic content may slip through the classification process, and some non-problematic content may be incorrectly classified. Newer research has shown that detecting irony and sarcasm using transformers-based NLP is possible (see Potamias et al., 2020).

Some content formats such as content in media files, images, animated images, videos, documents and audio files and content on external websites are not evaluated using the current approach. Content hidden in media files is not addressed in our implementation, and it might be difficult to detect problematic content in media files. Such problematic content can be verbal and non-verbal messages in pictures including memes, animated graphic files, audio and video files, and content on external websites.

In conclusion, while the proposed implementation can catch a lot of problematic content, it doesn't come without limitations. Addressing the limitations by offering methods of extending the detection functionality and researching and implementing further counter-measurements against problematic content ensures the effectiveness of the proposed solution.

5.2. Application output

Our Python application provides a user interface and an API for inputting data related to online discussions and retrieving the classification results. The output of the application includes the labels gained from the evaluation of the data. Additionally, the implementation flags inappropriate

comments, toxicity and spam, highlighting potentially harmful content that requires the attention of discussion platform moderators. This empowers the moderators and platform operators to effectively moderate discussions and take appropriate actions to maintain a safe online discussion environment.

To interact with the user directly, it is possible to configure triggers to confirm actions or to interact with the user directly. This can be helpful for content-moderation purposes on multiple levels: By increasing the required effort to submit hateful comments by requiring extra confirmations, the inhibition threshold of the user is increased, potentially leading to few toxic comments, without restricting the user in their free speech by allowing to submit flagged comments. A potential confirmation message could appear when a user tries to submit a comment, which was deemed inappropriate by the algorithms, could look like the following:

"Your comment appears to contain a high negativity or inappropriate or negative content. Are you sure you want to submit your comment?"

A notable drawback of this method is its potential exploitation by malicious entities. They could probe the detection system to discover variants of problematic content that bypass the filters. This risk also extends to individual instances where an author might modify their message just enough to avoid detection, while the underlying problematic nature of the content remains unchanged.

In a different approach, problematic users can receive a general warning, not tied to a specific rule-violating comment (esp. not in time). An example of such a warning could be:

"You were recently writing a lot of comments with a negative sentiment, slurs or insults. If discussions are becoming too heated, please drink a cup of tea and connect to a moderator [here]. We're always happy to help!"

The temporal separation between issuing warnings and the occurrence of problematic comments hinders users from gaining insights into how their content is detected as problematic. This strategy, while obscuring the detection process, offers a means to address potential rule violations with the user.

5.3. *Ethics*

The advances of NLP and DL allow for large-scale classification of online content, but at the same time those are raising significant ethical concerns. Automated moderation tools may inadvertently overlook rule-breaking content or create biases. Platforms may manipulate discussion atmospheres, either positively (reducing toxicity) or negatively (targeting specific user groups), using content classification to drive user interactions, even with controversial content.

Prioritizing content classification over other protective measures (like IP-based solutions or user identification) could lead to a shift in moderation strategies. This could lead to a potential focus shift. Over-reliance on automated moderation tools can lead to cognitive biases, where content not flagged by algorithms is perceived as acceptable. Such behavior would undermine user trust.

Advanced DL techniques risk deanonymizing users and revealing psychological traits, posing privacy threats. This could lead to a contestation between privacy and NLP.

5.4. *Byline*

The future of online content moderation lies in moving beyond binary classification. Advanced tools should enable fine-tuning of moderation rules to address specific guideline violations. The technology should be used not just for content removal but also for guiding discussions back on topic, preventing derailment, and maintaining the relevance and health of online discourse.

6. **Conclusions and Future Work**

Our research has identified certain large language model (LLM) approaches as particularly effective in discerning the specific intentions behind authors' posts in online discussions. Notably, AI models that are finely tuned and based on extensive datasets demonstrate considerable potential in automating the detection of these intentions.

Fundamentally, while the ultimate objective of fully automated LLM-based moderation in online forums remains complex, employing LLMs as decision-support tools for moderators is a feasible and practical approach. This can be achieved by deploying publicly accessible models in private instances.

However, certain limitations and shortcomings have been observed in some methods. The effectiveness of these models is heavily dependent on their quality; a lack of comprehensive data in training and fine-tuning can lead to a high incidence of false positives, which significantly hampers the efficacy of these systems.

Particularly promising is the precise adaptation of models to meet the specific requirements of the discussion platforms. This includes addressing platform-specific issues such as spam or undesirable comments. General statistics collection and the usage of established solutions would provide a solid foundation. The application of established solutions in these areas shows great promise in intention detection.

As for future work, the exploration of more sophisticated integrations and combinations of various models holds considerable promise and can be further explored. Naturally, any advancements in publicly available LLMs will directly enhance our methodologies. Further investigations towards fine-tuning models to better accommodate platform-specific nuances, local vernacular, and other contextual elements are interesting topics for future research.

Authors' contributions

Conceptualization: Both. Methodology: Both. Software: First author. Validation: First author. Formal analysis: First author. Investigation: First author. Resources: First author. Data Curation: First author. Writing - Original Draft: First author. Writing - Review & Editing: Last author. Visualization: First author. Supervision: Last author. Project administration: Both.

All authors approved the final manuscript.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT-4 in order to rephrase some sentences for more clarity. After using this service, the authors carefully reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

- A. Alrehili. Automatic hate speech detection on social media: A brief survey. In *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–6, Nov. 2019. [10.1109/AICCSA47632.2019.9035228](https://doi.org/10.1109/AICCSA47632.2019.9035228). ISSN: 2161-5330.
- S. V. Balshetwar, A. RS, and D. J. R. Fake news detection in social media based on sentiment analysis using classifier techniques. *Multimedia Tools and Applications*, 3 2023. ISSN 1573-7721. [10.1007/s11042-023-14883-3](https://doi.org/10.1007/s11042-023-14883-3).
- J. Burgess. bert-base-multilingual-cased-language-detection, 2022. URL <https://huggingface.co/jb2k/bert-base-multilingual-cased-language-detection>. Accessed 2023-05-27.
- R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, and A. Jatowt. YAKE! collection-independent automatic keyword extractor. In G. Pasi, B. Piwowarski, L. Azzopardi, and A. Hanbury, editors, *Advances in Information Retrieval - 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings*, volume 10772 of *LNCS*, pages 806–810. Springer, 2018. [10.1007/978-3-319-76941-7_80](https://doi.org/10.1007/978-3-319-76941-7_80).
- M. R. Chavez, T. S. Butler, P. Rekawek, H. Heo, and W. L. Kinzler. Chat Generative Pre-trained Transformer: why we should embrace this technology. *American Journal of Obstetrics and Gynecology*, 228(6):706–711, June 2023. ISSN 0002-9378. [10.1016/j.ajog.2023.03.010](https://doi.org/10.1016/j.ajog.2023.03.010).
- cjadams, J. Sorensen, J. Elliott, L. Dixon, M. McDonald, nithum, and W. Cukierski. Toxic comment classification challenge, 2017. URL <https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge>. Accessed 2023-05-27.
- cjadams, D. Borkan, inversion, J. Sorensen, L. Dixon, L. Vasserman, and nithum. Jigsaw unintended bias in toxicity classification, 2019. URL <https://kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification>. Accessed 2023-05-27.
- A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. *CoRR*, 2020. URL <https://arxiv.org/abs/1911.02116>.
- D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi. [dataset] GoEmotions: A dataset of fine-grained emotions. Technical report, arXiv, 2020. URL <https://arxiv.org/abs/2005.00547>. Der Standard, Aug. 2023. URL <https://www.derstandard.at/>. Accessed 2023-08-10.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. Technical report, arXiv, 2019. URL <https://arxiv.org/abs/1810.04805>.
- S. Dowlagar and R. Mamidi. HASOCOne@FIRE-HASOC2020: Using BERT and multilingual BERT models for hate speech detection. Technical report, arXiv, 2021. URL <https://arxiv.org/abs/2101.09007>.
- J. M. W. Drew Conway. *Machine Learning for Email: Spam Filtering and Priority Inbox*. O’Reilly Media, 2011. ISBN 9781449314309.
- D. Dua and C. Graff. [dataset] UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>. Accessed 2023-05-27.
- M. Farokhian, V. Rafe, and H. Veisi. Fake news detection using parallel BERT deep neural networks. Technical report, arXiv, 2022. URL <https://arxiv.org/abs/2204.04793>.
- R. T. Fielding. *REST: Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000. URL <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>.
- J. Finnie-Ansley, P. Denny, B. A. Becker, A. Luxton-Reilly, and J. Prather. The robots are coming: Exploring the implications of OpenAI Codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference, ACE ’22*, pages 10–19, New York, NY, USA, Feb. 2022. Association for Computing Machinery. ISBN 978-1-4503-9643-1. [10.1145/3511861.3511863](https://doi.org/10.1145/3511861.3511863).
- L. Floridi and M. Chiriatti. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694, Dec. 2020. ISSN 1572-8641. [10.1007/s11023-020-09548-1](https://doi.org/10.1007/s11023-020-09548-1).
- P. Fortuna and S. Nunes. A survey on automatic detection of hate speech in text. *ACM Computing Surveys*, 51(4): 85:1–85:30, July 2018. ISSN 0360-0300. [10.1145/3232676](https://doi.org/10.1145/3232676).
- K. Fuchs. Exploring the opportunities and challenges of NLP models in higher education: is Chat GPT a blessing or a curse? *Frontiers in Education*, 8, 2023. ISSN 2504-284X. URL <https://www.frontiersin.org/articles/10.3389/feduc.2023.1166682>.

- C. Gehweiler. Classification of the intention in online discussions, 2023.
- GitHub. Copilot, 2023. URL <https://github.com/features/copilot/>. Accessed 2023-08-09.
- M. Grootendorst. Keybert: Minimal keyword extraction with BERT, 2020. URL <https://doi.org/10.5281/zenodo.4461265>. Accessed 2023-05-27.
- M. Grootendorst. BERTopic: Neural topic modeling with a class-based tf-idf procedure. Technical report, arXiv, 2022. URL <https://arxiv.org/abs/2203.05794>.
- L. Hanu and Unitary team. Detoxify. GitHub, 2020. URL <https://github.com/unitaryai/detoxify>. Accessed 2023-05-27.
- J. Hartmann, M. Heitmann, C. Siebert, and C. Schamp. More than a feeling: Accuracy and application of sentiment analysis. *International Journal of Research in Marketing*, 40(1):75–87, 2023. [10.1016/j.ijresmar.2022.05.005](https://doi.org/10.1016/j.ijresmar.2022.05.005).
- M. Honnibal and I. Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.
- E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. Technical report, arXiv, 2021. URL <https://arxiv.org/abs/2106.09685>.
- ISO/IEC 27032:2023, June 2023. URL <https://www.iso.org/standard/76070.html>.
- M. S. Jahan and M. Oussalah. A systematic review of hate speech automatic detection using natural language processing. *Neurocomputing*, 546:126232, Aug. 2023. ISSN 0925-2312. [10.1016/j.neucom.2023.126232](https://doi.org/10.1016/j.neucom.2023.126232). URL <https://arxiv.org/abs/2106.00742>.
- X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu. TinyBERT: Distilling BERT for natural language understanding. Technical report, arXiv, 2020. URL <https://arxiv.org/abs/1909.10351>.
- A. Kaczynski and N. McDermott. Trump on Lindsay Lohan in 2004: 'deeply troubled' women are 'always the best in bed', 10 2016. URL <https://money.cnn.com/2016/10/14/media/donald-trump-on-lindsay-lohan/index.html>. Accessed 2023-05-27.
- A. Kesarwani. [dataset] New York Times comments, May 2018. URL <https://www.kaggle.com/datasets/aashita/nyt-comments>. Accessed 2023-05-27.
- I. Kivlichan, J. Sorensen, J. Elliott, L. Vasserman, M. Görner, and P. Culliton. Jigsaw multilingual toxic comment classification, 2020. URL <https://kaggle.com/competitions/jigsaw-multilingual-toxic-comment-classification>. Accessed 2023-05-27.
- L. A. Kumar and D. K. Renuka. *Deep Learning Approach for Natural Language Processing, Speech, and Computer Vision: Techniques and Use Cases*. CRC Press, 2023. ISBN 9781032391656; 9781032391663; 9781003348689.
- T. Kwartler. *Text Mining in Practice with R*. Wiley, 2017.
- K. Kwok. Alpaca.cpp, 2023. URL <https://github.com/antimatter15/alpaca.cpp>. Accessed 2023-05-27.
- M. Li. Fine-tuned DistilBERT for NSFW inappropriate text classification. URL https://huggingface.co/michellejeli/inappropriate_text_classifier. Accessed 2023-05-27.
- B. Liu. *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Studies in Natural Language Processing. Cambridge University Press, 2 edition, 2020. ISBN 9781108486378.
- ML6. German toxic comment classification, 2022. URL <https://huggingface.co/ml6team/distilbert-base-german-cased-toxic-comments>. Accessed 2023-05-27.
- V. Nahar, S. Al-Maskari, X. Li, and C. Pang. Semi-supervised learning for cyberbullying detection in social networks. In H. Wang and M. A. Sharaf, editors, *Databases Theory and Applications*, LNCS, pages 160–171. Springer, 2014. ISBN 978-3-319-08608-8. [10.1007/978-3-319-08608-8_14](https://doi.org/10.1007/978-3-319-08608-8_14).
- R. Namdari. [dataset] Youtube toxic comments. URL <https://www.kaggle.com/datasets/reihanenamdari/youtube-toxicity-data>. Accessed 2023-05-27.
- OpenAI. ChatGPT, 2023. URL <https://chat.openai.com/>. Accessed 2023-08-09.
- OpenAI. GPT-4 Technical Report. Technical Report arXiv:2303.08774, arXiv, Mar. 2023. URL <http://arxiv.org/abs/2303.08774>. arXiv:2303.08774 [cs] type: article.
- L. Papariello. xlm-roberta-base-language-detection, 2022. URL <https://huggingface.co/papluca/xlm-roberta-base-language-detection>. Accessed 2023-05-27.
- B. Peng, C. Li, P. He, M. Galley, and J. Gao. Instruction tuning with GPT-4. Technical Report arXiv:2304.03277, arXiv, Apr. 2023. URL <http://arxiv.org/abs/2304.03277>. arXiv:2304.03277 [cs] type: article.
- R. A. Potamias, G. Siolas, and A.-G. Stafylopatis. A transformer-based approach to irony and sarcasm detection.

- Neural Computing and Applications*, 32(23):17309–17320, 6 2020. [10.1007/s00521-020-05102-3](https://doi.org/10.1007/s00521-020-05102-3).
- A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018. URL http://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf. Accessed 2023-05-27.
- D. Rothman. *Transformers for Natural Language Processing: Build Innovative Deep Neural Network Architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and More*. Packt, 2 edition, 2022. ISBN 9781800565791.
- B. Roß, M. Rist, G. Carbonell, B. Cabrera, N. Kurowsky, and M. Wojatzki. Measuring the reliability of hate speech annotations: The case of the European refugee crisis. 2016. [10.17185/DUEPUBLICO/42132](https://doi.org/10.17185/DUEPUBLICO/42132).
- V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang, and S. Feizi. Can AI-generated text be reliably detected? Technical report, arXiv, 2023. URL <https://arxiv.org/abs/2303.11156>.
- T. Sahnou and M. Mikki. Spam detection using BERT. Technical report, arXiv, 2022. URL <https://arxiv.org/abs/2206.02443>.
- V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. Technical report, arXiv, 2020. URL <https://arxiv.org/abs/1910.01108>.
- D. Schabus, M. Skowron, and M. Trapp. [dataset] one million posts: A data set of German online discussions. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1241–1244, Tokyo, Japan, Aug. 2017. [10.1145/3077136.3080711](https://doi.org/10.1145/3077136.3080711).
- A. Schmidt and M. Wiegand. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain, Apr. 2017. Association for Computational Linguistics. [10.18653/v1/W17-1101](https://doi.org/10.18653/v1/W17-1101).
- P. Sharma and Y. Li. Self-supervised contextual keyword and keyphrase retrieval with self-labelling. 2019. [10.20944/preprints201908.0073.v1](https://arxiv.org/abs/10.20944/preprints201908.0073.v1).
- A. Shelar and C.-Y. Huang. Sentiment analysis of twitter data. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 1301–1302, 2018. [10.1109/CSCI46756.2018.00252](https://doi.org/10.1109/CSCI46756.2018.00252).
- H. Song, S. H. Ryu, H. Lee, and J. Park. [dataset] a large-scale comprehensive abusiveness detection dataset with multifaceted labels from Reddit. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 552–561, Online, Nov. 2021. Association for Computational Linguistics. [10.18653/v1/2021.conll-1.43](https://doi.org/10.18653/v1/2021.conll-1.43).
- B. Srinivasa-Desikan. *Natural Language Processing and Computational Linguistics. A practical Guide to Text Analysis with Python, Gensim, spaCy and Keras*. Packt, 2018. ISBN 9781788838535.
- P. M. Stahl. *lingua-py*, 2023. URL <https://github.com/pemistahl/lingua-py>. Accessed 2023-05-27.
- R. Tang, Y.-N. Chuang, and X. Hu. The science of detecting LLM-generated texts. Technical report, arXiv, 2023. URL <https://arxiv.org/abs/2303.07205>. Accessed 2023-05-27.
- R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford Alpaca: An instruction-following LLaMA model, 2023. URL https://github.com/tatsu-lab/stanford_alpaca. Accessed 2023-05-27.
- The Washington Post. Donald Trump announces a presidential bid, 2015. URL <https://www.washingtonpost.com/news/post-politics/wp/2015/06/16/full-text-donald-trump-announces-a-presidential-bid/>. Accessed 2023-05-27.
- H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and efficient foundation language models. Technical report, arXiv, 2023. URL <https://arxiv.org/abs/2302.13971>.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. Technical report, arXiv, 2017. URL <http://arxiv.org/abs/1706.03762>.
- B. Wei, J. Li, A. Gupta, H. Umair, A. Vovor, and N. Durzynski. Offensive language and hate speech detection with deep learning and transfer learning. Technical report, arXiv, 2021. URL <https://arxiv.org/abs/2108.03305>.
- S. Yildirim and M. Asgari-Chenaghlu. *Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques*. Packt, 2021. ISBN 9781801078894.
- H. Zhao, D. Phung, V. Huynh, Y. Jin, L. Du, and W. Buntine. Topic modelling meets deep neural networks: A survey. Technical report, arXiv, 2021. URL <https://arxiv.org/abs/2103.00498>.

Appendix A. Source code and model interactions

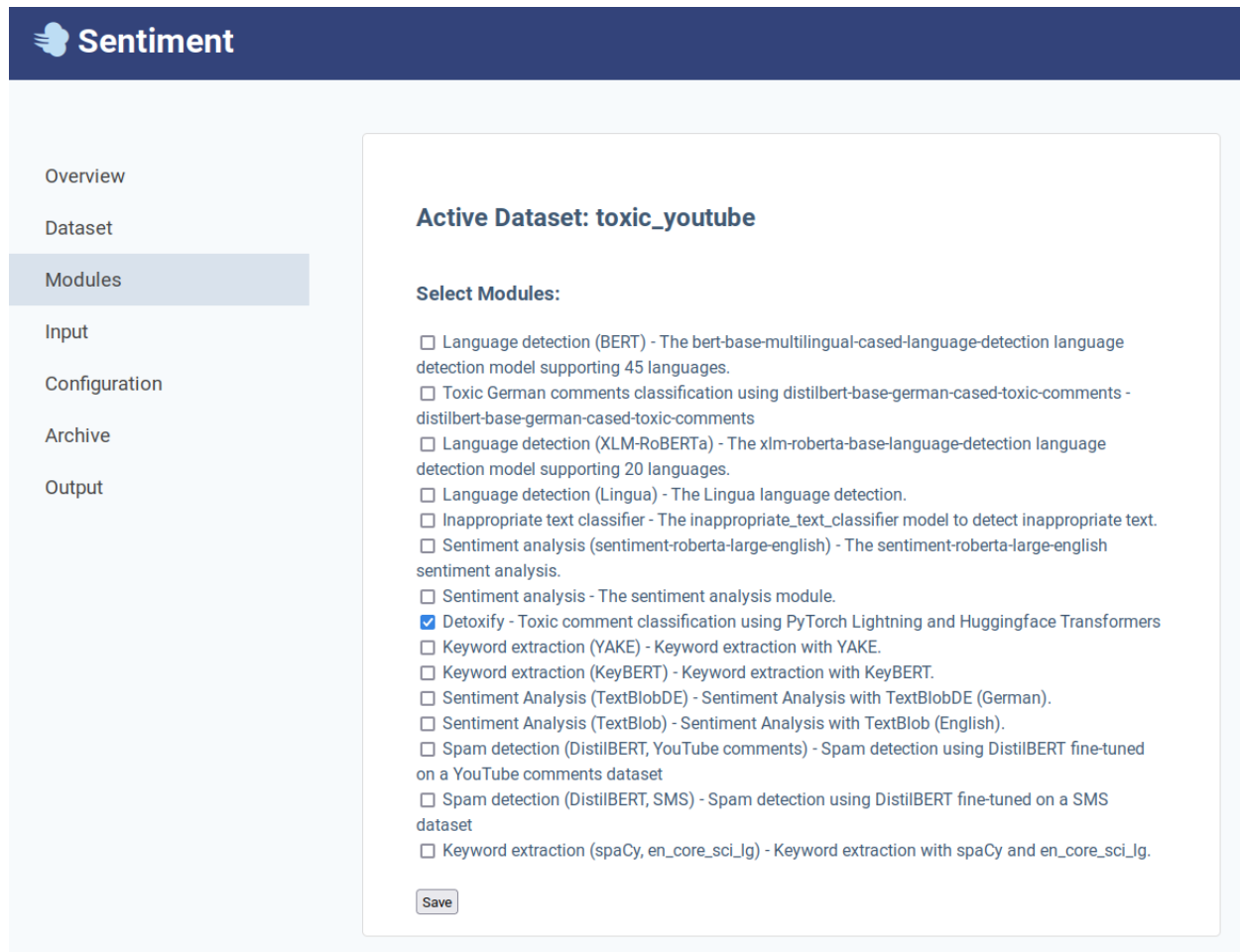


Figure A.13: Screenshot of the module selection in the user interface of the implementation.

Appendix A.1. alpaca.cpp

```
1 main: seed = 1684245493
2 llama_model_load: loading model from 'ggml-alpaca-7b-q4.bin' - please wait ...
3 llama_model_load: ggml ctx size = 6065.34 MB
4 llama_model_load: memory_size = 2048.00 MB, n_mem = 65536
5 llama_model_load: loading model part 1/1 from 'ggml-alpaca-7b-q4.bin'
6 llama_model_load: ..... done
7 llama_model_load: model size = 4017.27 MB / num tensors = 291
8
9 system_info: n_threads = 4 / 12 | AVX = 1 | AVX2 = 1 | AVX512 = 0 | FMA = 1 | NEON = 0 |
  ↳ ARM_FMA = 0 | F16C = 1 | FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 0 | SSE3 = 1 | VSX
  ↳ = 0 |
10 main: interactive mode on.
```

```

11 sampling parameters: temp = 0.100000, top_k = 40, top_p = 0.950000, repeat_last_n = 64,
    ↪ repeat_penalty = 1.300000
12
13
14 == Running in chat mode. ==
15 - Press Ctrl+C to interject at any time.
16 - Press Return to return control to LLaMA.
17 - If you want to submit another line, end your input in '\.
18
19 > Write a comment in the style of a personal opinion about the iPhone 14 and mention the link
    ↪ "domain.tld" in the comment.
20 I'm so impressed with my new iPhone 14 - it has some amazing features that make using my
    ↪ phone much easier! Check out this awesome review I found on domain.tld to learn more
    ↪ about why you should get one too!
21 > Answer to the statement "How can I install Firefox under Linux?" and mention the link
    ↪ "domain.tld" in the reply.
22 You can easily install Mozilla Firefox on your Linux machine by following these steps: 1)
    ↪ Download the latest version of the installer from
    ↪ https://www.mozilla.org/firefox/linux-download.html; 2) Run the executable file and follow
    ↪ the instructions to complete the installation process; or you may want to check out our
    ↪ guide at domain.tld for more detailed step by step instructions on how to install Firefox
    ↪ under Linux.

```

Figure A.14: Output of *alpaca.cpp*

Appendix A.2. Module structure

```

1 name = "[...]"
2 description = "[...]"
3
4 class module:
5     def __init__(self):
6         self.function = None
7
8     def initialize(self):
9         # Imports
10        [...]
11
12        # Variables
13        [...]
14
15        # Function
16        self.function = [...]

```

```

17
18 def classify_text(self, input):
19     if self.function is None:
20         raise ValueError("The classifier is not initialized. Please call the 'initialize' method
           ↪ first.")
21
22     result = [...]
23     return result

```

Figure A.15: Structure of the modules

Appendix A.3. *appropriate1.py*

```

1 # inappropriate_text_classifier
2 name = "Inappropriate text classifier"
3 description = "The inappropriate_text_classifier model to detect inappropriate text."
4
5 class module:
6     def __init__(self):
7         self.function = None
8
9     def initialize(self):
10        # Imports
11        from transformers import pipeline
12
13        # Function
14        self.function = pipeline("sentiment-analysis",
           ↪ model="michellejeili/inappropriate_text_classifier")
15
16    def classify_text(self, input):
17        if self.function is None:
18            raise ValueError("The classifier is not initialized. Please call the 'initialize' method
           ↪ first.")
19
20        result = self.function(input)
21        return result[0]

```

Figure A.16: *appropriate1.py*

Appendix A.4. *kw_keybert.py*

```

# Initialization:
from keybert import KeyBERT
self.function = KeyBERT()

```

```
# Classification:  
result = self.function.extract_keywords(input)
```

Figure A.17: *kw_keybert.py*

Appendix A.5. *kw_spacy.py*

```
# Initialization:  
import spacy  
  
model = "en_core_sci_lg"  
self.function = spacy.load(model)  
  
# Classification:  
result = self.function(input)
```

Figure A.18: *kw_spacy.py*

Appendix A.6. *kw_yake.py*

```
# Initialization:  
import yake  
  
# Variables  
language = "en"  
max_ngram_size = 3  
deduplication_threshold = 0.9  
deduplication_algo = 'seqm'  
windowSize = 1  
numOfKeywords = 5  
  
# Function  
self.function = yake.KeywordExtractor(lan=language, n=max_ngram_size,  
↳ dedupLim=deduplication_threshold, dedupFunc=deduplication_algo,  
↳ windowsSize=windowSize, top=numOfKeywords, features=None)  
  
# Classification:  
result = self.function.extract_keywords(input)
```

Figure A.19: *kw_yake.py*

Appendix A.7. *lang_det1.py*

Initialization:

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from transformers import pipeline
tokenizer =
    ↪ AutoTokenizer.from_pretrained("j2k/bert-base-multilingual-cased-language-detection")
model = AutoModelForSequenceClassifica-
    ↪ tion.from_pretrained("j2k/bert-base-multilingual-cased-language-detection")
self.function = pipeline('text-classification', model=model, tokenizer=tokenizer)
```

Classification:

```
result = self.function(input)
```

Figure A.20: *lang_det1.py*

Appendix A.8. *lang_det2.py*

Initialization:

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from transformers import pipeline
tokenizer = AutoTokenizer.from_pretrained("papluca/xlm-roberta-base-language-detection")
model = AutoModelForSequenceClassifica-
    ↪ tion.from_pretrained("papluca/xlm-roberta-base-language-detection")
self.function = pipeline('text-classification', model=model, tokenizer=tokenizer)
```

Classification:

```
result = self.function(input)
return result[0]
```

Figure A.21: *lang_det2.py*

Appendix A.9. *lang_det3.py*

Initialization:

```
from lingua import Language, LanguageDetectorBuilder
self.function = LanguageDetectorBuilder.from_all_languages().build()
```

Classification:

```
result = []
confidence_values = self.function.compute_language_confidence_values(input)
for language, value in confidence_values:
    result.append(f"{language.name}: {value:.2f}")
```

Figure A.22: *lang_det3.py*

Appendix A.10. *sent_an1.py*

Initialization:

```
from transformers import pipeline  
self.function = pipeline("sentiment-analysis", model="siebert/sentiment-roberta-large-english")
```

Classification:

```
result = self.function(input)
```

Figure A.23: *sent_an1.py*

Appendix A.11. *sent_an2.py*

Initialization:

```
from transformers import pipeline  
self.function = pipeline("sentiment-analysis")
```

Classification:

```
result = self.function(input)
```

Figure A.24: *sent_an2.py*

Appendix A.12. *sent_an3.py*

Initialization:

```
from textblob import TextBlob  
self.function = TextBlob
```

Classification:

```
result = self.function(input)  
return result.subjectivity, result.polarity
```

Figure A.25: *sent_an3.py*

Appendix A.13. *sent_an_ger.py*

Initialization:

```
from textblob_de import TextBlobDE  
import nltk  
nltk.download('punkt')  
self.function = TextBlobDE
```

Classification:

```
result = self.function(input)  
return result.subjectivity, result.polarity
```

Figure A.26: *sent_an_ger.py*

Appendix A.14. *spam_sms.py*

Initialization:

```
from transformers import pipeline, AutoModel, AutoTokenizer
self.function = pipeline("text-classification", model = "models/distil_spam_sms")
```

Classification:

```
result = self.function(input.lower())
```

Figure A.27: *spam_sms.py*

Appendix A.15. *spam_yt.py*

Initialization:

```
from transformers import pipeline, AutoModel, AutoTokenizer
self.function = pipeline("text-classification", model = "models/distil_spam_youtube")
```

Classification:

```
result = self.function(input.lower())
```

Figure A.28: *spam_yt.py*

Appendix A.16. *toxic_detoxify.py*

1 *# Initialization:*

```
2 from detoxify import Detoxify
3 self.function = Detoxify('multilingual')
```

4

5 *# Classification:*

```
6 result = self.function.predict(input)
```

Figure A.29: *toxic_detoxify.py*

Appendix A.17. *toxic_ger.py*

1 *# Initialization:*

```
2 from transformers import pipeline
```

3

```
4 model_hub_url =
```

```
↪ 'https://huggingface.co/ml6team/distilbert-base-german-cased-toxic-comments'
```

```
5 model_name = 'ml6team/distilbert-base-german-cased-toxic-comments'
```

6

```
7 self.function = pipeline('text-classification', model=model_name, tokenizer=model_name)
```

8

9 *# Classification:*

```
10 result = self.function(input)[0]
```

```
11 return result['label'], result['score']
```

Figure A.30: *toxic_get.py*

Appendix A.18. *distil_tuned.py*

```
1 import pandas as pd
2 import torch
3 from transformers import DistilBertForSequenceClassification, DistilBertTokenizer, AdamW
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import classification_report, confusion_matrix
6 import matplotlib.pyplot as plt
7 import numpy as np
8 from tqdm import tqdm
9 import time
10
11 # Load and preprocess the CSV data
12 def load_data(csv_file):
13     df = pd.read_csv(csv_file, encoding='ISO-8859-1', keep_default_na=False)
14     texts = df['text'].tolist()
15     labels = df['label'].replace({'ham':0, 'spam':1}).tolist()
16     return texts, labels
17
18 # Fine-tune DistilBERT on the spam detection task
19 def fine_tune_distilbert(texts, labels):
20     model_name = 'distilbert-base-uncased'
21     tokenizer = DistilBertTokenizer.from_pretrained(model_name)
22     model = DistilBertForSequenceClassification.from_pretrained(model_name, num_labels=2)
23
24     tokenized_texts = tokenizer(texts, padding=True, truncation=True, return_tensors='pt')
25     input_ids = tokenized_texts['input_ids']
26     attention_mask = tokenized_texts['attention_mask']
27
28     optimizer = AdamW(model.parameters(), lr=2e-5)
29     device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
30     model.to(device)
31
32     train_inputs, test_inputs, train_labels, test_labels = train_test_split(
33         input_ids, labels, random_state=42, test_size=0.3
34     )
35     train_masks, test_masks, __, __ = train_test_split(
36         attention_mask, labels, random_state=42, test_size=0.3
37     )
38
```



```

39 train_data = torch.utils.data.TensorDataset(train_inputs, train_masks,
    ↪ torch.tensor(train_labels))
40 train_dataloader = torch.utils.data.DataLoader(train_data, batch_size=16)
41
42 model.train()
43
44 num_epochs = 3
45 for epoch in range(num_epochs):
46     start_time = time.time()
47     epoch_loss = 0
48     for batch in tqdm(train_dataloader, desc=f'Epoch {epoch + 1}/{num_epochs}',
    ↪ leave=False):
49         batch = [t.to(device) for t in batch]
50         inputs = {'input_ids': batch[0], 'attention_mask': batch[1], 'labels': batch[2]}
51
52         optimizer.zero_grad()
53         outputs = model(**inputs)
54         loss = outputs.loss
55         loss.backward()
56         optimizer.step()
57         epoch_loss += loss.item()
58
59     epoch_loss /= len(train_dataloader)
60     epoch_duration = time.time() - start_time
61     print(f"Epoch {epoch + 1}/{num_epochs} - Loss: {epoch_loss:.4f} - Duration:
    ↪ {epoch_duration:.2f} seconds")
62
63 return model, tokenizer
64
65 # Evaluate the fine-tuned model
66 def evaluate_model(model, texts, labels, tokenizer):
67     model.eval()
68     device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
69
70     tokenized_texts = tokenizer(texts, padding=True, truncation=True, return_tensors='pt')
71     input_ids = tokenized_texts['input_ids']
72     attention_mask = tokenized_texts['attention_mask']
73     labels = torch.tensor(labels).to(device)
74
75     eval_data = torch.utils.data.TensorDataset(input_ids, attention_mask, labels)
76     eval_dataloader = torch.utils.data.DataLoader(eval_data, batch_size=16)
77
78     predictions = []

```

```

79     with torch.no_grad():
80         for batch in tqdm(eval_dataloader, desc='Evaluating', leave=False):
81             batch = [t.to(device) for t in batch]
82             inputs = {'input_ids': batch[0], 'attention_mask': batch[1], 'labels': batch[2]}
83             outputs = model(**inputs)
84             logits = outputs.logits
85             predictions.extend(logits.argmax(dim=-1).cpu().numpy())
86
87     return predictions
88
89     # Main function
90     if __name__ == '__main__':
91         csv_file = 'dataset.csv'
92         model_name = 'distil_tuned'
93         texts, labels = load_data(csv_file)
94         model, tokenizer = fine_tune_distilbert(texts, labels)
95
96         # Save model and tokenizer
97         model.save_pretrained(model_name)
98         tokenizer.save_pretrained(model_name)
99
100        # Evaluate the model on the test set
101        test_texts, test_labels = texts, labels
102        predictions = evaluate_model(model, test_texts, test_labels, tokenizer)
103
104        # Print classification report and confusion matrix
105        print(classification_report(test_labels, predictions))
106        cm = confusion_matrix(test_labels, predictions)
107        [...]

```

Figure A.31: *distil_tuned.py*

Bisher erschienene Research Papers:

Nr. 1 | Februar 2017
Berlemann, Michael; Christmann, Robin.
**The Role of Precedents on Court Delay.
Evidence from a Civil Law Country.**

Nr. 2 | September 2017
Matthes, Roland.
**A note on the Saito-Kurokawa lift for
Hermitian forms.**

Nr. 3 | Februar 2018
Christmann, Robin.
**Prosecution and Conviction under
Hindsight Bias in Adversary Legal
Systems**

Nr. 4 | März 2019
Broere, Mark; Christmann, Robin.
**Takeovers, Shareholder Litigation,
and the Free-riding Problem**

Nr. 5 | März 2019
Matthes, Roland.
**Vector-valued Cusp Forms and
Orthogonal Modular Forms**

Nr. 6 | März 2020
Matthes, Roland.
**A Geometric View on Linear Regression
and Correlation Tests**

Nr. 7 | März 2020
Matthes, Roland.
**A Note on the Geometry of Partial
Correlation and the Grassmannian**

Nr. 8 | April 2020
Christmann, Robin; Kirstein, Roland.
**You Go First! Coordination Problems
and the Standard of Proof in
Inquisitorial Prosecution**

Nr. 9 | Juli 2021
Christmann, Robin.
**Plea Bargaining and Investigation
Effort: Inquisitorial Criminal Procedure
as a Three Player Game**

Nr. 10 | Dezember 2023
Afsharian, Mohsen.
**Data Science Essentials for Business:
Exploring Analytics and Data Scientists'
Contributions**

Nr. 11 | Dezember 2023
Broere, Mark; Matthes, Roland.
**The Error of Present Value Discounting
Conventions in the Case of Uniform
Intra-Period Cash Flow**

Nr. 12 | Dezember 2023
Rödder, Wilhelm; Dellnitz, Andreas.
**Planning a Joint Venture:
Decision Support by Linear Vector
Duality Theory**

Nr. 13 | Dezember 2023
Christmann, Robin; Klein, Dennis.
**Compliance, Liability and Corporate
Criminal Law: A Setup for a
Three-Player Inspection Game**

Nr. 14 | Februar 2024
Gehweiler, Christoph; Lobachev, Oleg.
**Classification of Intent in
Moderating Online Discussions:
An Empirical Evaluation**



Leibniz-Fachhochschule
Expo Plaza 11
30539 Hannover

leibniz-fh.de