The 14th IEEE International Conference on Dependable Systems, Services and Technologies, DESSERT'2024
11-13 October, 2024, Athens, Greece

# Effective Content Moderation Using Modern AI Tools

Kateryna Bortnyk
*Department of Computer Engineering and Security, Lutsk National Technical University*
Lutsk, Ukraine
katerina.bortnyk@gmail.com

Nataliia Bahniuk
*Department of Computer Engineering and Security, Lutsk National Technical University*
Lutsk, Ukraine
bahniuk_nataliia@lutsk-ntu.com.ua

Inna Kondius
*Dean of the Faculty of Computer and Information Technologies, Lutsk National Technical University,*
Lutsk, Ukraine
Innastk@ukr.net

Kateryna Melnyk
*Department of Computer Engineering and Security, Lutsk National Technical, University.*
Lutsk, Ukraine
ekaterinamelnik@gmail.com

Yuliia Melnychuk
*Department of Digital Educational Technologies, Lutsk National Technical University,*
Lutsk, Ukraine
cezar61mv@gmail.com

Kostiantyn Kondius
*Department of Software Engineering, Lutsk National Technical University,*
Lutsk, Ukraine
kondius_kostya@ukr.net

***Abstract*— In the paper we develop a web application that integrates AI capabilities for content moderation, while providing an effective and balanced user protection system based on general comparative analysis of two leading tools for content moderation: Microsoft Content Moderator and Google Perspective API. Moderation mechanisms using AI algorithms have been developed, allowing not only to filter unwanted content, but also to adapt to changing conditions and user needs. We present the comparison results of two language models of Google Perspective API and Microsoft Content Moderator and appropriate tools taking into account the specific helpful, honest, harmless criteriums that is aligned with human values and include emotional tonality supported by developers.**

***Keywords*— *web application, artificial intelligence, content moderation, software***

## I. INTRODUCTION

Moderation of online content is nowadays an integral part of ensuring the safety and quality of Internet platforms using. But with the growth of content diversity traditional moderation methods are insufficient. In this context, artificial intelligence (AI) can be used, which can improve the content moderation process with aid of its analytical and computational abilities.

These research questions are videly researched besause AI systems are the replication of human intelligence processes by technology, and can be used for example in the social media field, as shown in [1]. AI systems allow machines to learn from experience, adapt to new inputs, and execute all human-like tasks resulted in substantial breakthroughs in computer performance in tasks that were previously exclusively attainable for humans [2, 3]. The article [4] is devoted to the actual challenges in use of AI for content moderation poses to law. Recent researches [5,6] in this field provide the promise of freeing humans from repetitive tasks by AI, thus enabling the humankind to focus on more creative actions. One of the examples of automating traditionally human-deal tasks is content moderation relating desinformation and explicitly harmful content, for example in social platforms [7-9]. Recently human reviewers monitored and detected any content that violated the communities standards [10]. That said, the high volume of posts in current platforms coupled with the advances in AI has led platforms to automate their content moderation, harnessing the responsiveness of AI. But now significant development of AI would be used for content moderation, since new challenges such as detection accuracy, ethical dilemmas, and data privacy are constantly emerging.

In this context, this paper is devoted to thoroughly explore the opportunities and challenges of integrating artificial intelligence with content moderation in web applications since one of the most important aspects of the modern Internet ecosystem is the control and moderation of content published by users. Due to the scale and variety of content that is generated and shared on the network every day, the task of moderation becomes extremely difficult and requires high efficiency. The aim of this paper is to develop a web application that integrates AI capabilities for content moderation, while providing an effective and balanced user protection system based on general comparative analysis of two leading tools for content moderation: Microsoft Content Moderator and Google Perspective API. In this context we develop moderation mechanisms using AI algorithms, allowing not only to filter unwanted content, but also to adapt to changing conditions and user needs. We give the comparison research results of two language models of Google Perspective API and Microsoft Content Moderator and appropriate tools taking into account the specific helpful, honest, harmless criteriums that is aligned with human values and include emotional tonality supported by developers.

This paper is organized as following. The basic concepts and research methodology design, specification and system requirements are presented in section II and III. In section IV we present web application design with content moderation using AI, and in section V and VI we present system development and specification, test results and discussions. Conclusion and recommendation for a future work are presented in last section.

## II. STATE-OF-THE-ART

The research problem posted in this paper is devoted to development of web application with online chat for the content moderation system using artificial intelligence (AI). Creating a web application with online chat is a necessary prerequisite for the implementation of highly effective content moderation with the help of artificial intelligence. High-tech solutions are implemented, maximum security and user comfort in the web environment is ensured. This

approach contributes to the creation of an innovative and modern application that meets the requirements of modern web standards and meets the needs and expectations of users, providing them with a reliable and pleasant interaction in the online environment.

The history of the development of content moderation shows constant adaptation to growing challenges and new trends in the online environment [11], besause modern artificial intelligence technologies play an important role in facilitating the content moderation process. Therefore machine learning algorithms and neural networks can automatically detect unwanted content based on various criteria, as shown in [12]. For example, they can recognize images with offensive content or analyze text for violations. However, despite the advances in the field of artificial intelligence, the task of moderation remains difficult due to the diversity of content and the rapid development of new methods of bypassing filters. Therefore, experienced moderators and automated systems work in cooperation to ensure the quality and efficiency of moderation [13-16]. Despite the achievements in the field of artificial intelligence, moderation remains a difficult task, the same content can be interpreted differently depending on the context, which creates the need for manual review by moderators. There is also the risk of content being mistakenly deleted, which can lead to conflicts and user complaints.

The novelty of this research lies in the integration of improved AI algorithms that provide more efficient and accurate content moderation, while preserving data privacy and avoiding censorship. The work includes an analysis of the current state of this field, examples of successful integration of AI into this process, future prospects for content moderation with the integration of artificial intelligence, and draws conclusions about the potential advantages and challenges of this approach.

## III. Basic Concepts And Research Methodology

### *A. Technological solutions for the integration of AI*

Choosing a programming language and framework for the server and front-end is a key step in designing an information platform. Productivity, efficiency and development capabilities depend on this choice.

#### *a) Google Perspective API*

The Google Perspective API was chosen due to its high level of accuracy and efficiency in detecting toxic content in textual form, it is based on machine learning and has a wide functionality for text analysis. An important advantage of the Google Perspective API is its integration with different languages and ease of use of artificial intelligence can be used to improve the quality of content moderation in web application. The integration of artificial intelligence using the Google Perspective API allows to reduce the dependence on intervention in the moderation process, which makes this process faster, more accurate and objective.

The Google Perspective API supports the use of several programming languages, including Python, which is one of the most popular languages among web application developers. Python stands out for its convenience and flexibility, as well as rich libraries for working with machine learning and text analysis. This makes it an ideal choice for integration with tools such as the Google Perspective API.

Python has a rich ecosystem of libraries for machine learning and data analysis, such as TensorFlow, Keras, PyTorch, Pandas, and Scikit-learn, which can be used to enhance the functionality of web applications.

#### *b) Microsoft Content Moderator*

In this work we use Microsoft Content Moderator - an innovative cloud service created by Microsoft, it uses advanced artificial intelligence technologies to detect potentially offensive, dangerous or unwanted content in a variety of formats such as text, image and video files.

We use it for effectively perform the following tasks:

1. Detection of offensive content: to recognize different forms of discrimination, such as racism, sexism and homophobia, ensuring a high level of objectivity in content moderation.

2. Dangerous Content Detection: to identify threats of violence, terrorist propaganda, and other forms of extremism, helping to keep users safe.

3. Unwanted Content Detection: to identify advertising, spam, and other forms of irrelevant or unnecessary material, making content filtering easier.

The integration of Microsoft Content Moderator into a web application with online chat opens wide perspectives for solving important tasks:

1. Improving the security and comfort of users: ensuring automatic detection and blocking of offensive or dangerous content, reducing the risk of interaction with harmful expressions.

2. Reducing the burden on moderators: automatic detection and filtering of most content, freeing up human resources for more complex scenarios.

3. Reducing costs for content moderation: optimization of work processes and resources, which contributes to effective moderation and cost savings.

Usind it we  we achieve the following goals: ensuring the safety and comfort of users: the use of advanced technologies to create a safe and pleasant environment for interaction; reducing the burden on moderators: improving productivity and optimizing work processes for effective moderation; reduction of costs for content moderation: rational use of resources and the opportunity to save funds allocated for moderation.

## IV. Design Specification and System Requirements

### *A. Creation of a message verification mechanism*

We develop a safe and comfortable environment for online chat users, providing content moderation to detect and block offensive, infringing and unwanted chat messages.

Functional requirements for the chat include possibility of creating user accounts and authorization in the system, sending and receiving text messages in real time, the possibility of creating group chats and private dialogues, notification of new messages, possibility to delete or edit own messages, chat moderation  for preventing the spread of offensive, aggressive and disruptive content, ensuring a high level of safety and comfort for users, compliance with the rules and regulations of the web application community.

Creating a message verification mechanism *includes*:

1. Development of a software module to transfer text messages sent by users to an API for analysis.

2. Setting the parameters of requests to Google Perspective API and Microsoft Content Moderator to determine toxicity, spam, offensive content and other negative elements in the text.

3. Optimization of the mechanism for speed and accuracy of analysis.

During the development of the software module for the transfer of text messages to the API, the high efficiency and reliability of this process we take into account. API request parameters has to be configured to detect different aspects of negative content, including toxicity, allowing for more accurate detection and handling of unwanted content.

The Python programming language is used to implement the message verification mechanism to interact with the Google Perspective and Microsoft Content Moderator APIs. The Django web framework is used for the convenience of processing incoming text messages and transferring them to the specified API for analysis.

### *B. System reaction to detected toxic content*

The system response to detected toxic content includes:

1. Development of an algorithm to determine how the system should respond to detected toxic messages. This may include blocking the message, warning the user, or sending a message to a moderator.

2. Setting different severity levels for responding to toxic content, from mild to severe violation of community rules.

To ensure an effective response to detected toxic content, an algorithm is developed that will take into account the severity of the violation and comply with established community rules. This allows the system to respond adequately and ensure the safety of users.

To process and respond to detected toxic content, Python has to be used to develop an algorithm that will determine how the system should respond to each specific message. If needed, the Django Channels library can be used to implement asynchronous operations, such as reacting to toxic content in real time. This library allows to efficiently implement asynchronous operations in Django using advanced WebSockets technology. WebSockets technology, in turn, provides an opportunity to establish a two-way communication channel between the web browser and the server. This opens up opportunities for instant, real-time data sharing, enabling fast and efficient response to events such as the detection of toxic content. This approach allows maintaining a high level of security and comfort for users in time-sensitive web scenarios.

### *C. Interfaces Specification*

We use Django to develop an interface where users can monitor the moderation status of their messages and receive notifications about blocking or warnings. The use of Python and the Django REST Framework will allow to effectively organize data exchange between the server part and the user interface, ensuring optimal system functionality.

Features of the interface for administrators and moderators include:

1. Development of a separate interface for administrators and moderators, which will provide them with the means to effectively detect and manage unwanted content.

2. Defining capabilities for administrators, including the ability to block users, delete messages, and set rules for moderation.

The interface for administrators (Fig. 1) and moderators will be designed taking into account their needs in detecting and managing unwanted content. It will provide them with the tools to effectively control chat and message moderation.

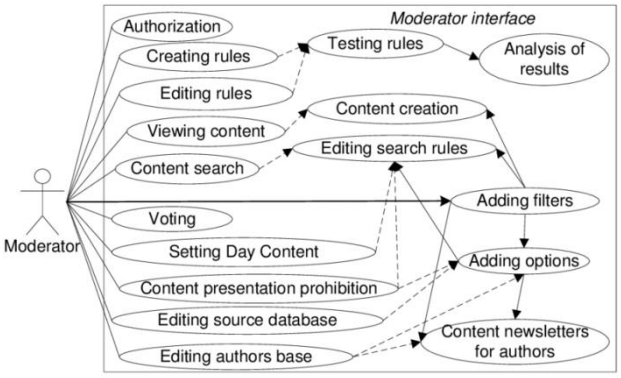


Fig. 1. Case diagram of content moderators interface

Django will be used to create a separate interface for admins and moderators where they can take action on unwanted content. Python and the Django Admin library will provide the ability to manage users, messages and set moderation rules.

An ordinary *User* has the right to send messages, view messages from other users, and also receive notifications about the results of moderation.

A *Moderator* has additional rights, such as blocking users or deleting toxic messages.

An *Administrator* get advanced system management capabilities, including assigning and managing moderators, setting moderation rules, integrating with APIs, and monitoring the overall chat status.

This system of roles and rights will be implemented by the Django web framework through authorization and access control mechanisms, allowing each user to perform their duties and have access to appropriate functions depending on their role in the system.

### *D. Online chat design and functionality*

In this section, we consider the design and functionality of an online chat with a content moderation system, which has to be implemented using the integration of Google Perspective API and Microsoft Content Moderator. The main goal is to create an intuitive and convenient interface for users and moderators who will interact with the system.

Administrators and moderators have their own interface to effectively manage the moderation system, where they can not only view but also take action on all messages. They also be able to take action against unwanted content and notify users when messages are blocked or deleted.

To implement this functionality, we use Django web framework in combination with a set of libraries for user interface development. Django Templates is used to create the HTML pages of the interface, and the Django Forms library is used to handle input from users and moderators.

### *E. Security and Privacy in AI-Moderated Online Chat*

We consider security and privacy aspects of online chat with a content moderation system based on Google

Perspective API and Microsoft Content Moderator integration. Special attention we pay to the protection of personal data of users and prevention of possible abuse or attacks on the system. To ensure the confidentiality of user data, all data transmitted between users and the server is encrypted using modern cryptographic protocols such as SSL/TLS. This will provide protection against data interception. In order to effectively combat spam, flooding and other forms of abuse in the chat, we implement a filtering and blocking system to identify and terminate excessively active users. A limit is set on the frequency of sending messages, aimed at preventing flooding and other forms of abuse. This measure allows monitoring and regulating user activity, increasing the overall level of security and comfort in the chat. For the successful implementation of the specified security measures, the built-in tools of the Django web framework, recognized for its reliability and ability to guarantee a high level of protection is used. Thanks to these means, the system's resistance to potential threats and abuses is ensured. In addition, we use appropriate libraries for data encryption and filtering. This allows not only to effectively encrypt confidential information, but also to improve data filtering for the purpose of timely detection and prevention of possible threats. Such a comprehensive approach to ensuring system security allows creating a reliable barrier against potential attacks and provide a high degree of information protection. Full and complex security of the system as a whole has to be implemented.

## V. System Development and Specification

### A. Backend Development Using Python and Django

Python and Django are key components in back-end development of modern web applications. Python, with its readable syntax and wide range of libraries, provides the basis for efficient and flexible backend development. Being a high-level framework written in Python, Django allows developers to quickly build secure and reliable web applications by providing powerful tools for query processing, data modeling, user authentication, and session and message management.

The development of the server part includes the creation of models that define the structure of the database and corresponding views that handle the logic of requests and responses. Django ORM (Object-Relational Mapping) allows developers to work with the database using Python code without resorting to writing SQL queries directly, which improves performance and security.

Django also has built-in support for database migrations, making it easy to make changes to the database structure without the risk of data loss. In addition, Django provides mechanisms for form processing, data validation and security, such as protection against CSRF attacks. Gunicorn is used as a WSGI HTTP server for deploying Django applications. It acts as a bridge between a web server (for example, Nginx) and a Django application, providing parallel processing of requests and increasing the efficiency of the application.

The user model extends Django's built-in AbstractUser model by adding additional fields that allow you to store detailed information about each user. This model includes standard user attributes such as name, email, password, as well as specific fields such as profile photo, bio, and date of birth. The message model serves to store messages sent in the chat. It includes information about the sender, the recipient, the content of the message and the time of sending. Messages contain sender and recipient fields that are associated with the user model, as well as a message text field and a timestamp. The `is_read` field is used to track whether a message has been read. These models provide the basis for implementing the functionality of a web chat application, allowing for effective user interaction and message management.

PostgreSQL is a powerful, open and free relational database management system that is often used together with Django to build web applications. Before starting the integration, you need to install PostgreSQL on the development environment and on the server. After installation, you need to create a new database and a user account that Django will use to access the database.

In the `settings.py` file of the Django project, you need to make changes to the database configuration. Typically, this includes specifying the database name, user, password, and host (Listing 1).

**Listing 1.** Setting up a database connection in Django:

```
DATABASES = {
    "default": {
        "ENGINE":
"django.db.backends.postgresql",
        "NAME": "mydatabase",
        "USER": "mydatabaseuser",
        "PASSWORD": "mypassword",
        "HOST": "localhost",
        "PORT": "5432",   }}
```

Django ORM (Object-Relational Mapping) allows developers to interact with the database using Python objects, abstracting from the need to write SQL queries. This makes it easier to work with the database, provides security and keeps the code clean. Django uses a migration system to manage changes in the database structure. This allows you to safely make changes to models and display them in the database. The makemigrations and migrate commands are used to create and apply migrations. When working with a database, it is important to ensure security, especially when it comes to storing passwords and accessing the database. Django provides built-in tools for this, such as an authentication system and password encryption.

### B. Development of the server part of the chat using Django channels

Django Channels extends the capabilities of Django by making it easy to build asynchronous applications such as chats. Redis is used as a communication channel and intermediate data storage. Django Channels and Redis must be installed to start chat development. This can be done with pip: `pip install channels channels_redis`.

The next stage is the creation of `routing.py`. ASGI (Asynchronous Server Gateway Interface) routing is a key component in developing asynchronous web applications such as chats on Django Channels. The ASGI router defined in the `routing.py` file is responsible for routing incoming WebSocket connections to the appropriate consumers. This allows for the two-way communication between the client and the server that is necessary for instant messaging in chat.

*Redis*, as the backend for channels in Django Channels, plays an important role in ensuring efficient and fast transmission of messages between chat users. It functions as an intermediate message store, enabling scalable and reliable real-time data transfer. Using Redis helps maintain stability

and high availability of the chat system, even under heavy load conditions.

The *Consumer* in Django Channels plays the role of a controller that manages WebSocket connections. The consumers.py file (Listing 2) implements methods for processing connection and disconnection events and receiving messages from clients. Consumer uses asynchronous programming to provide non-blocking data processing, which is critical for high-performance chat systems where multiple connections are served simultaneously.

**Listing 2**. Example consumers.py:

```
import json
from channels.generic.websocket import
AsyncWebsocketConsumer
class ChatConsumer(AsyncWebsocketConsumer):
    async def connect(self):
        # Switch-on logic
        await self.accept()
    async def disconnect(self, close_code):
        # Logic of switch-off
        pass
    async def receive(self, text_data):
        # Message processing logic
        text_data_json = json.loads(text_data)
        message = text_data_json["message"]
        await
self.send(text_data=json.dumps({"message":
message}))
```

*C. Development of the client part of the web chat*

Connecting the client side to a websocket and implementing chat functionality requires integrating JavaScript, HTML, and CSS to create an interactive and responsive interface. The chat interface can be implemented using HTML and CSS. The developed system interface and identifacation process is done on Fig. 2,3.

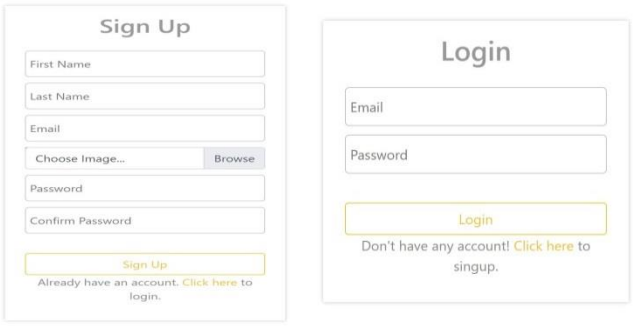


Fig. 2. The Identification Process of System

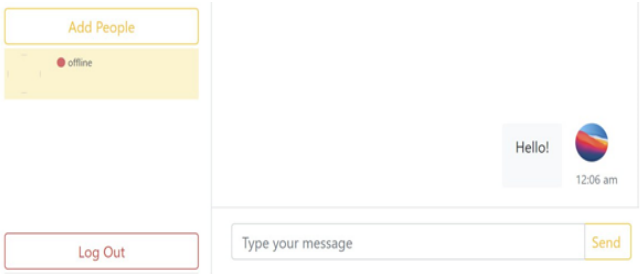


Fig. 3. The System Interface

The main template contains a text field for entering messages and an area for displaying chat history. CSS styles can be used to visually design the chat. JavaScript is used to connect to the server through websockets and exchange messages (Listing 3). After establishing a websocket connection, the client script can receive messages from the server and display them in the web interface. At the same time, the messages entered by the user are sent to the server for processing and distribution to other chat participants.

**Listing 3.** Example websocket listener:

```
document.addEventListener('DOMContentLoaded',
function () {
  var ws_scheme = window.location - protocol ==
"https:" ? "wss" : "ws";
  var chatSocket = new WebSocket(
    ws_scheme + '://' + window.location.host +
'/ws/chat/'   );
  document.querySelector('#chat-message-
submit').onclick = function (e) {
    var messageInputDom = document -
querySelector('#chat-message-input');
    var message = messageInputDom.value;
    chatSocket.send(JSON.stringify({
      'message': message
    }));
    messageInputDom.value = '';   };
  chatSocket.onmessage = function (e) {
    var data = JSON - parse(e.data);
    document.querySelector('#message-
list').innerHTML += ('‹p›' + data.message +
'‹/p›');   }; });
```

*D. Chat content moderation using Microsoft Content Moderator and Google Perspective API*

Moderation of content in chat applications includes the use of algorithms to automatically detect and filter unwanted content, as well as providing tools for manual moderation. This may include integration with external services such as Google Perspective API or Microsoft Content Moderator to improve the efficiency of moderation processes.

Microsoft Content Moderator provides an API for analyzing text and images to detect potentially unwanted content. To integrate this service into a Django application, we use the following algorithm and code (Listing 4):

1. To receive a text message from the user.
2. To send this message to the Microsoft Content Moderator API.
3. To get an answer with content analysis.
4. To check the result for the presence of undesired content.
5. To respond accordingly: allow the publication of the message, reject it or flag it for manual moderation.

**Listing** 4. Example of algorithm implementation for moderation using Microsoft Content Moderator:

```
import requests
from django.conf import setting
def moderate_text_with_microsoft(text):
api_url=http://api.contentmoderator.microsoft.com
/
contentmoderator/moderate/v1.0/ProcessText/Screen
headers={
```

```
"Ocp-Apim-Subscription-Key":
setting.MICROSOFT_CONTENT_MODERATOR_API_KEY}
response_data=response.json()
if response_data["Terms"]:
return "the message contains unwanted content"
else:
return "the message is received"
```

Google Perspective API uses machine learning to assess the likelihood of toxicity in text messages. Code and algorithm:

1. Receiving a text message from the user.
2. Sending a message for analysis to the Google Perspective API.
3. Obtaining the result of the toxicity assessment from the API.
4. Determining whether the toxicity score exceeds a given threshold.
5. Deciding whether the publication of the message is allowed.

So, we have examined the key aspects of integrating artificial intelligence (AI) into the content moderation process, with a particular focus on the development of a web project that includes moderated chat. The integration of AI has proven to be extremely important to ensure the safety and comfort of users in the virtual environment, as well as to enforce the rules of the communities. The project's use of artificial intelligence for chat moderation highlights its ability to effectively detect and manage unwanted content, including offensive content and rule violations.

The moderated web chat project is an example of a successful integration of AI into a real web application. This confirms the importance of intelligent systems for ensuring the safety and quality of virtual communication. The future of content moderation with the integration of AI promises to remain a dynamic and innovative field, where new technologies and solutions are aimed at improving the efficiency and accuracy of content moderation.

## VI. Experiments Results and Assessments

The best way to test the quality of various systems is to calculate the assessment rate. According to the introducing assessment, we can also test the different language models in the systems and dictionary size. So, we have developed a tool used to test these models in Google Perspective API or Microsoft Content Moderator. Also, we have calculated the relative assessments by using this tool to estimate a list of sentences, which we collected in the form of separate 15 files (see Table I). We do these steps to test Google Perspective API and Microsoft Content Moderator by comparison of the efficiency. Algorithm for detecting abusive content in audio content include main following steps: input data; check according to the dictionary of words with abusive content; test text content; definition of abusive content; calculation of assessments of the contents; output data and make a decision on the acceptability of content according to the level of abuse. To detect abusive content in language text content we identify each component that indicates the presence of certain abuse options, that is effectively detected by using of abusive words. The input data for identifying abusive content in researching text collected into studied files is a dictionary of abusive speech words and test content of the files. The first step is to analyze the content expressed in a textual representation using dictionary. The abusive assessment (AA) of the content is determined statistically and is calculated according to the formula:

$$\theta = \frac{\beta}{\rho}$$

where $\beta$ is the number of abusive words used in the text representation of the content, which are contained in the dictionary of abuse words, Paremeter $\rho$ is the total number of words in the content, $\theta$ is abusive coefficient or assessment of abuse. We define general weight assessment of abuse (GWAA) by the following way:

$$\gamma = \frac{k_1\theta + k_2 p}{2}$$

where abusive assessment (AA) $\theta$ is a statistical abusive content component, $p$ is a toxicity assessment (TA) given by native speaker estimators assessment which estimate using the certain criteriums [15] including helpful, honest, harmless that is aligned with human values and emotional tonality of the converted audio content. k1, k2 – weighting coefficients of the importance of indicators, that are elected experimentally, but $k_1 + k_2 = 1$. This human-based estimation is applied taking into account the typical practic of Google Perspective API developers [14]. For each comment, 3-10 raters who speak the appropriate language comment on whether the comment contains an attribute "toxicity", defined as $p$ following the given instructions. For each comment, 3-10 raters who speak the appropriate language comment on whether the comment contains an attribute "toxicity", defined as p following the given instructions. We then post-process the annotations to obtain the proportion of estimators who labeled the comment as "toxic", that determines the estimation $p$.

The output data is a numerical assessment of the abusiveness of the content and a conclusion on the acceptability of the content based on the level of abuse. The content acceptability conclusion based on the level of abuse is formed based on the threshold value set by the expert. Thus, the proposed algorithm for detecting abusive manifestations in the studied content allows you to obtain a numerical assessment of the abusiveness of the content and a conclusion on its acceptability.

TABLE I. Test Results and Comparison

| File number | Microsoft Content Moderator | | | Google Perspective API | | |
|---|---|---|---|---|---|---|
| | $\theta$ | γ | p | $\theta$ | γ | p |
| 1 | 0.3 | 0.2 | 0.1 | 0.25 | 0,18 | 0.1 |
| 2 | 0.4 | 0.5 | 0.6 | 0.52 | 0.56 | 0.6 |
| 3 | 0.9 | 0.85 | 0.8 | 0.82 | 0.81 | 0.8 |
| 4 | 1 | 0.9 | 0.8 | 0.85 | 0.82 | 0.8 |
| 5 | 0.7 | 0.6 | 0.5 | 0.6 | 0.55 | 0.5 |
| 6 | 0.2 | 0.35 | 0.5 | 0.4 | 0.45 | 0.5 |
| 7 | 0.8 | 0.85 | 0.9 | 0.83 | 0.86 | 0.9 |
| 8 | 0.8 | 0.9 | 1 | 0.93 | 0.96 | 1 |
| 9 | 0.1 | 0.25 | 0.4 | 0.2 | 0.3 | 0.4 |
| 10 | 0.3 | 0.45 | 0.6 | 0.68 | 0.64 | 0.6 |
| 11 | 0.2 | 0.35 | 0.5 | 0.45 | 0.47 | 0.5 |
| 12 | 0.7 | 0.75 | 0.8 | 0.77 | 0.78 | 0.8 |
| 13 | 0.4 | 0.45 | 0.5 | 0.47 | 0.48 | 0.5 |
| 14 | 0.9 | 0.85 | 0.8 | 0.87 | 0.83 | 0.8 |
| 15 | 0.1 | 0.2 | 0.3 | 0.25 | 0.27 | 0.3 |

Using the available MatLab tools we calculate the values of GWAA assessments for or Microsoft Content Moderator (Fig.4-5) and Google Perspective API (Fig. 6-7) respectively. In order for predicting and analyze the unknown values we make the mathematical modeling using MatLab appropriate tools. On the Fig.5,7 the plotting of the datasets is presented. The data frame of file numbers is [0-15]. For this purpose we make fit in curve of the loaded data

using the appropriate Curve Fitting Toolbox, and obtain the polynomial for interpolation and characterize data with aid of a global fit in order to obtain a simple empirical models.

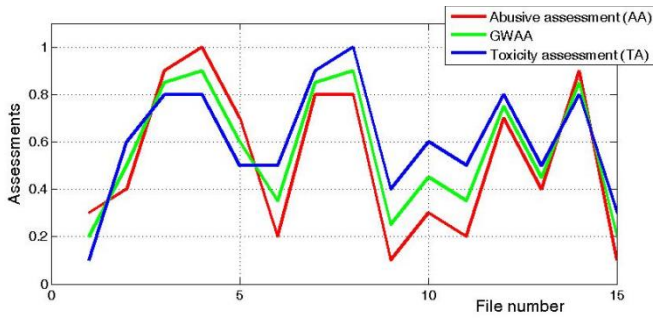


Fig. 4. Assessments for Microsoft Content Moderator

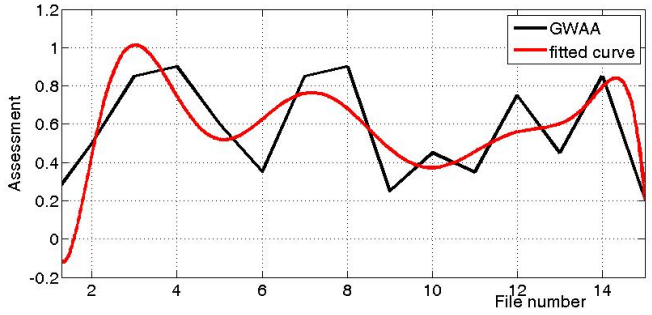


Fig. 5. GWAA assessment and relative empirical model for Microsoft Content Moderator

The main advantages of selected polynomial fit is reasonable flexibility for data. Analyzing the data-set (Fig.5) we obtain appropriate resulting linear model which is a polynomial of the degree 9 in a following form:

$$F(x) = p_1*x^9 + p_2*x^8 + p_3*x^7 + p_4*x^6 + p_5*x^5 + p_6*x^4 + p_7*x^3 + p_8*x^2 + p_9*x + p10$$

Coefficients (with 95% confidence bounds):

$p_1$ = -9.941e-07 (-2.914e-06, 9.259e-07); $p_2$ =7.371e-05 (-6.465e-05, 0.0002121); $p_3$=-.002325 (-0.006561, 0.001911); $p_4$ = 0.04066 (-0.03117, 0.1125); $p_5$ =-0.4299 (-1.167, 0.3075); $p_6$ = 2.815 (-1.888, 7.518); $p_7$ = -11.21 (-29.6, 7.172); $p_8$ = 25.58 (-16.15, 67.31); $p_9$ = -29.19 (-77.88, 19.5); $p_{10}$ = 12.61 (-8.876, 34.09).

The calculating efficiency are following: SSE: 0.3211; RSQUARE: 0.6764; DFE: 5; ADJRSQUARE: 0.0939; RMSE: 0.2534.

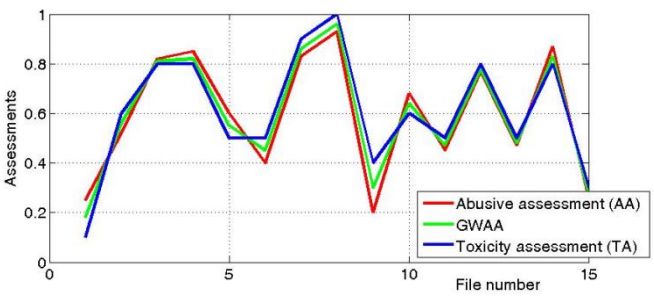


Fig.6. Assessments for Google Perspective API

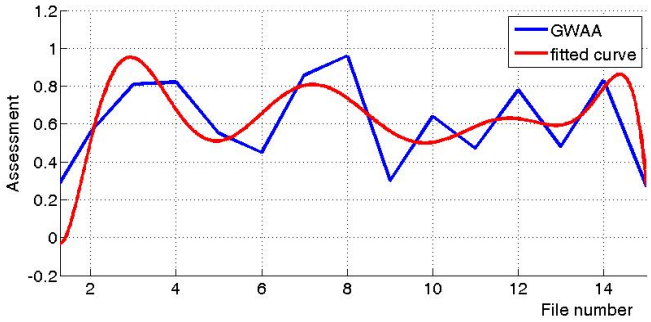


Fig. 7. GWAA assessment and relative empirical model for Google Perspective API

The relative empirical model (Fig.7) for Google Perspective API obtained by use of MATLAB/Curve Fitting Toolbox has a form:

$$F(x) = p_1*x^9 + p_2*x^8 + p_3*x^7 + p_4*x^6 + p_5*x^5 + p_6*x^4 + p_7*x^3 + p_8*x^2 + p_9*x + p10$$

Coefficients (with 95% confidence bounds): $p_1$ = -9.793e-07 (-2.772e-06, 8.135e-07); $p_2$=7.186e-05 (-5.732e-05, 0.0002011); $p_3$ = -0.002241 (-0.006196, 0.001714); $p_4$ = 0.03869 (-0.02838, 0.1058); $p_5$ = -0.4032 (-1.092, 0.2853); $p_6$ = 2.598 (-1.794, 6.989); $p_7$ = -10.16 (-27.33, 7.01); $p_8$ = 22.67 (-16.29, 61.64); $p_9$ = -25.23 (-70.69, 20.23); $p_{10}$ = 10.67 (-9.389, 30.73),

with the efficiency: SSE: 0.2799; RSQUARE: 0.6543; DFE: 5; DJRSQUARE: 0.0319; RMSE: 0.2366.

Based on these research we concluded that the tool that we have built to test the Google Perspective API or Microsoft Content Moderator by using some text files with real comments that were selected from different places with the original sentences showed that Google Perspective API achieved better efficiency, than Microsoft Content Moderator, as shown on Fig. From the graphics it can be seen that the difference between the rating curves is greater for Microsoft Content Moderator. Therefore, it can be concluded that the language model and appropriate tool of Google is better because of using helpful, honest, harmless criterium [15]. Our native speakers estimations based on human opinion that is aligned with human values and include emotional tonality have less difference for Google Perspective API than for Microsoft Content Moderator, as shown on Fig.6 comparing to results presented on Fig.4.

We present a general comparative (Table II) analysis of two leading tools for content moderation: Microsoft Content Moderator and Google Perspective API.

TABLE II. COMPARISON OF MICROSOFT CONTENT MODERATOR AND GOOGLE PERSPECTIVE API

| **Parameter** | **Microsoft Content Moderator** | **Google Perspective API** |
|---|---|---|
| Processing speed | High, but depends on the type of content | High for text queries |
| The accuracy of the analysis | High, with the possibility of training | High for toxic comments |
| Scalability | Suitable for large and small projects | Scales well for text data |
| Integration | Integrates with various platforms | Easily integrates with websites |
| Settings | Flexible for different types of content | Filters and toxicity indicators |
| Language support | Multilingual support | Multilingual, but with limitations |
| Expanding functionality | Support for text, images, video | Focus on textual analysis |

In our work the choice of Google Perspective API and Microsoft Content Moderator is justified by their exceptional accuracy and extensive set of features aimed at detecting unethical content.

The use of Google Perspective API machine learning to analyze the tone and semantics of texts is complemented by advanced filtering, spam and offensive content detection capabilities from Microsoft Content Moderator. Such comprehensive integration is seen as the source of an effective and stable moderation mechanism, contributing to the safety and convenience of users when interacting in online chat.

The use of Google Perspective API and Microsoft Content Moderator in online chat results in improved security and privacy for users. By using machine learning to detect and filter unwanted content such as hate speech, discrimination and violence, both APIs provide effective protection against potential harm and guarantee the privacy of users' personal data.

The benefit is improved customer service, as both APIs help chatbots provide more accurate and useful information, allowing them to distinguish between offensive or inaccurate questions. In addition, this integration helps reduce costs, as both APIs allow chatbots to handle more requests on their own, resulting in more efficient and cost-effective user service.

## VII. CONCLUSIONS

The conclusions of this work emphasize the relevance and importance of using automated systems in content moderation. Artificial intelligence, as evidenced by the development of a web project with chat, is a necessary tool for ensuring the quality and efficiency of moderation processes in online environments. It not only improves the speed of detection of unwanted content, but also opens up prospects for further development of this industry.

Based on presented research we concluded that Google Perspective API achieved better efficiency, than Microsoft Content Moderator. From the experimental results it can be seen that the difference between the rating curves is greater for Microsoft Content Moderator. Therefore, the language model and appropriate tool of Google is better because of using helpful, honest, harmless criterium based on human opinion that is aligned with human values and include emotional tonality.

The development of a web project with a chat and its successful integration with AI systems shows the need to create a safe and comfortable online environment for all users. Both tools provide important functionality in the field of digital moderation, but at the same time have significant differences in their approaches and areas of application. Microsoft Content Moderator covers a wide range of content types, including text, images, and video, and provides advanced customization and integration capabilities. On the other hand, the Google Perspective API focuses mainly on analyzing textual content, with a special emphasis on detecting toxic comments. When evaluated in the context of scalability, both tools demonstrate high performance and flexibility, making them suitable for a variety of applications from small to large data volumes. Such a comparison allows users to more consciously approach the choice of the appropriate tool depending on the specifics and requirements of their project.

## REFERENCES


[1] V.U. Gongane, M.V. Munot, & A.D. Anuse. “Detection and moderation of detrimental content on social media platforms: current status and future directions”. Soc. Netw. Anal. Min. no. 12, 129, 2022.

[2] S. Kemp. “Digital 2022: Malaysia — DataReportal – Global Digital Insights”. Retrieved May 14, 2022, from https://datareportal.com/reports/digital-2022-malaysia.

[3] A. Drootin, “Community Guidelines: The Legal Implications of Workplace Conditions for Internet Content Moderators”. Fordham L. Rev., no. 90:1197, 2021.

[4] R. Dimitrova. “Artificial Intelligence in Content Moderation – Legal Challenges and EU Legal Framework”, 2022 10th International Scientific Conference on Computer Science (COMSCI), Sofia, Bulgaria, 2022, pp. 1-6, doi: 10.1109/COMSCI55378.2022.9912595.

[5] Y. Guy. “Will AI augment or replace workers?”, 2023. https://www.forbes.com/, sites/forbestechcouncil/2023/08/08/will-ai-augment-or-replace-workers/

[6] T. Lykouris. W. Weng “Learning to Defer in Content Moderation: The Human-AI Interplay”, arXiv:2402.12237v3 [cs.LG] 2 Jun 2024.

[7] Meta Platforms, Inc. Facebook community standards, 2023. https://transparency. fb.com/policies/community-standards/. Accessed on December 19, 2023

[8] “X Corp. The X rules”, 2023. https://help.twitter.com/en/rules-and-policies/ x-rules. Accessed on December 19, 2023

[9] M. Hildebrandt “The Adaptive Nature of Text-driven Law”, Journal of Cross-disciplinary Research in Computational Law, 2022., no.1(1).

[10] D. Zhuang, X. Zhang, S. Song, and S. Hooker. “Randomness in neural network training: Characterizing the impact of tooling”. In Proceedings of Machine Learning and Systems, vol. 4, pp. 316–336, 2022.

[11] Mojtaba Komeili, Kurt Shuster, and Jason Weston. Internet-augmented dialogue generation, 2021, arXiv:2107.07566.

[12] P. Friedl. “Dis/similarities in the design and development of legal and algorithmic normative systems: the case of Perspective API”, Law, Innovation and Technology, no. 15:1, pp. 25-59, DOI: 10.1080/17579961.2023.2184134, 2023.

[13] A. Arsht and D. Etcovitch “The Human Cost of Online Content Moderation”. Harvard Journal of Law & Technology. 2018, URL: https://jolt.law.harvard.edu/digest/the-human-cost-of-online-content-oderation?onwardjourney=584162_v1.

[14] https://developers.perspectiveapi.com/s/about-the-api-training-data?language=en_US

[15] A.Askell, Y.Bai, A.Chen, D.Drain, D.Ganguli, T.Henighan, A. Jones, N.Joseph, B.Mann, N. DasSarma, others. “A General Language Assistant as a Laboratory for Alignment”. arXiv:2112.00861[cs.CL], Subjects: Computation and Language; Machine Learning, https://doi.org/10.48550/arXiv.2112.00861.

[16] G. Iason. “Artificial intelligence, values, and alignment”, Minds and Machines, no. 30(3), pp. 411–437, Sep 2020. doi:10.1007/s11023-020-09539-2.