

August 26th 2021

Modergator: Automated Content Moderation for Hateful Mememes, Speech and Text

Korbinian Koch, Skadi Dinter, Katrin Caragiuli



17% of German youth aged 18-24
have been **personally affected**
by hate speech online*

*YouGov/Institut für Demokratie und Zivilgesellschaft (2019)



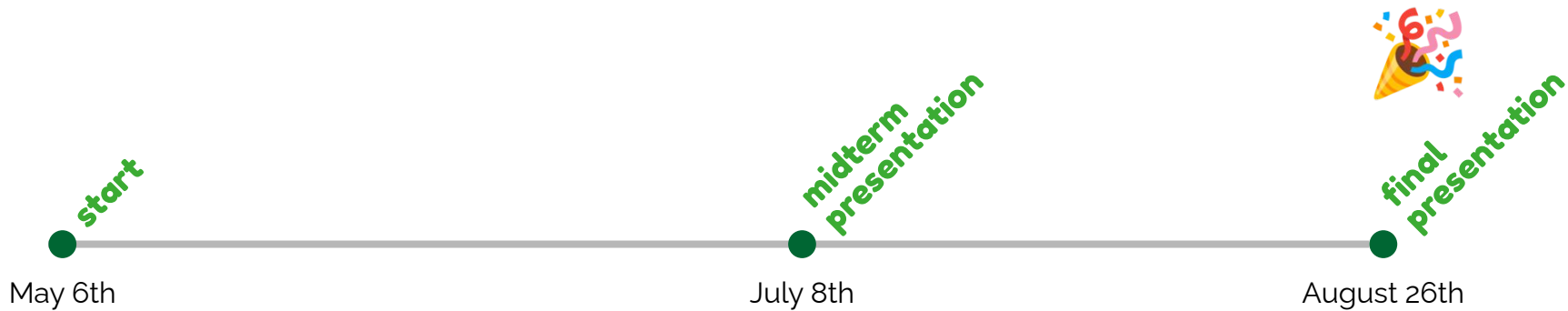
The Mission



Providing safer social spaces for everyone by making group moderation for hateful content easier and faster.

The Scope

- Master's project *Web Interfaces for Language Processing Systems*
- Team of 3(+) students
- duration of ~4 months



The Team



Korbinian

M.Sc. Intelligent Adaptive Systems

bot
functionality &

voice
processing

design

(and more)



Skadi

M.Sc. Informatics

meme model +
api

interactivity and
feedback

target group
detection

(and more)



Katrin

M.Sc. Informatics

meme model +
api

target group
detection

meme
detection api

(and more)

Modergator is ...

a collection of 6 APIs
revolving around hate
speech detection



curated components

and

a ready-to-use Telegram
bot moderating hateful
content in groups

modergator
bot

useable product

The Components



python-telegram-bot



[0,1]

Meme Detection
API

Contribution
from Niklas



OCR
API

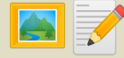
Contribution
from Niklas



[0,1]

Text Hatefulness
API

Contribution
from Fabian



[0,1]

Meme Hatefulness
API



Automatic Speech
Recognition
API



[Race, Sexuality, ...]

Target Group
API

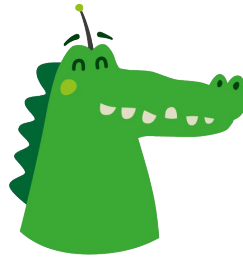
The Features

Automatic text classification:
hateful, offensive or normal



Analyze voice messages

GDPR-friendly opt-out



modergator
bot



Group members can discuss
the classification(s)

Detect affected target group(s)



Analyze memes

Published Project



Snapping back on hate speech

python v3.8 contributions welcome

[Key Features](#) • [How To Use](#) • [Installation](#) • [Components](#) • [Contributors](#) • [License](#)

👉 Modergator - Hate Detection for Text, Speech and Memes

Modergator is a Telegram bot able to moderate Telegram groups for hateful content.

Text messages are checked for whether they contain offensive and hateful speech, as well as the target groups that the speech is directed against (if there are any). Voice messages are transcribed and then handled the same way as a text messages.

Memes are also checked for hate which arises due to the combination of text and an image.

Currently, the bot can only understand English language.

🎯 Key Features

The bot will:

- check texts, voice messages and memes for hate
- intervene if necessary

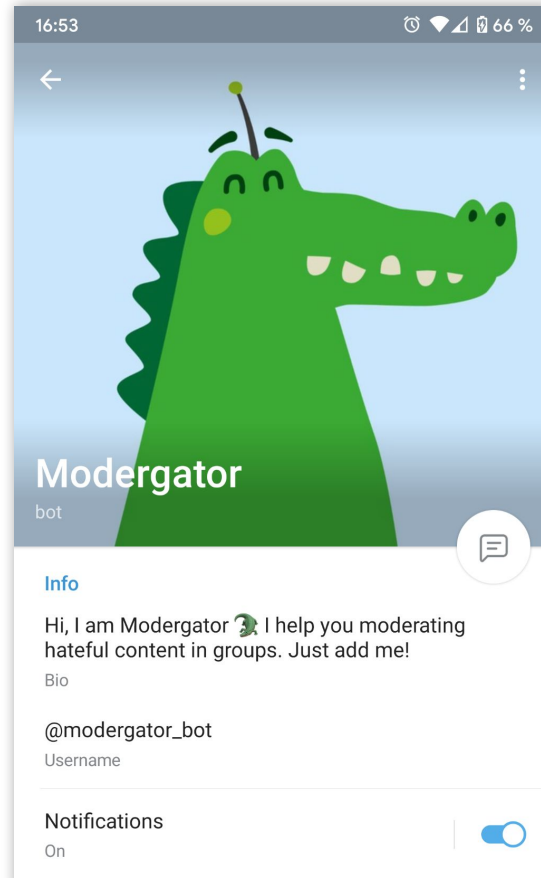
Group members can:

- dispute wrong classifications in a /poll
- optionally /optout of data processing (GDPR-compliant)

💡 How To Use

In order to interact with the bot, a Telegram account is needed. For instructions on how to create an account see: <https://telegram.org/>. To find

Published Bot



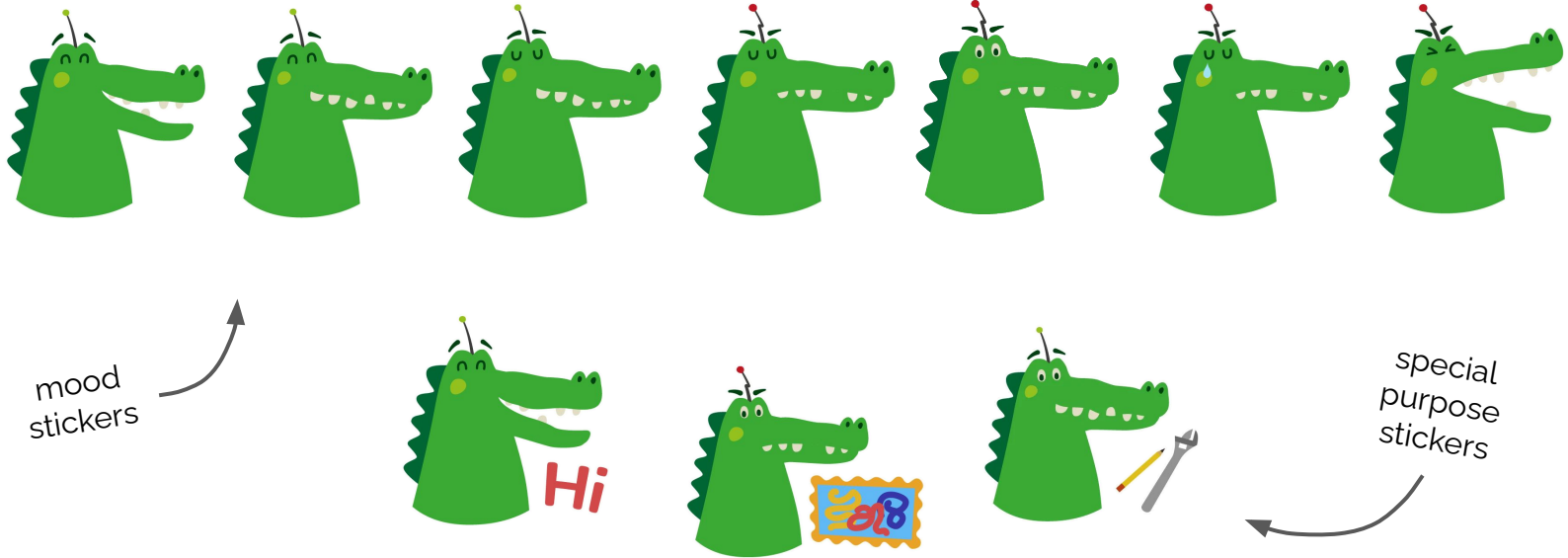
Design and UX: Name And Logo

modergator = portmanteau of moderator + alligator



Design and UX: Stickers

Custom Telegram Stickers to provide engaging user experience and create emotional connection



Technologies used



python-telegram-bot



Image sources: https://commons.wikimedia.org/wiki/File:Pytorch_logo.png <https://www.python.org/community/logos/> [https://en.wikipedia.org/wiki/Flask_\(web_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework))
https://www.startpage.com/av/proxy-image?piurl=https%3A%2F%2Fencrypted-tbn0.gstatic.com%2Fimages%3Fq%3Dtbn%3AANDgGcT_gVQL4e-dJj-rujJZk2ilAHhPfDZO-MUE4W7wo4IT14NMhlyo%26s&sp=1629920522T4bf13d0774a565978e0be1ffe5ab33a2f2b8a64a395e6c373a1d23a5e582cd0b <https://swagger.io/>
<https://github.com/python-telegram-bot/python-telegram-bot> <https://github.com/> <https://www.docker.com/company/newsroom/media-resources>
<https://github.com/Muennighoff/vilio>

Telegram Bot



- programmed in Python
- defines all commands such as
 - /help
 - /optout
 - ...
- and accesses the other APIs
- responsible for receiving and filtering messages
- as well as sending answers

```
def main() -> None:
    """Start the bot."""
    # Create the Updater and pass it your bot's token.
    updater = Updater(TOKEN)

    # Get the dispatcher to register handlers
    dispatcher = updater.dispatcher

    # on different commands - answer in Telegram
    dispatcher.add_handler(CommandHandler("start", start_command))
    dispatcher.add_handler(CommandHandler("about", about_command))
    dispatcher.add_handler(CommandHandler("help", help_command))
    dispatcher.add_handler(CommandHandler("optout", optout_command))
    dispatcher.add_handler(CommandHandler("joke", joke_command))
    dispatcher.add_handler(CommandHandler("howto", howto_command))
    dispatcher.add_handler(CommandHandler("poll", poll_command))
    dispatcher.add_handler(CommandHandler("optin", optin_command))
    dispatcher.add_handler(CommandHandler("debug", debug_command))
    dispatcher.add_handler(CommandHandler("goodvibes", goodvibes_command))
    dispatcher.add_handler(PollAnswerHandler(receive_poll_answer))
    dispatcher.add_handler(MessageHandler(Filters.poll, receive_poll))
    dispatcher.add_handler(MessageHandler(Filters.status_update.new_chat_members, welcome_message))

    # on non command i.e message - echo the message on Telegram
    dispatcher.add_handler(MessageHandler(Filters.text & ~Filters.command, handle_text))
    dispatcher.add_handler(MessageHandler((Filters.photo | Filters.document.category('image')) & ~Filters.command, handle_image))
    dispatcher.add_handler(MessageHandler(Filters.voice & ~Filters.command, handle_voice))
```

Target Detection



- Based on HateXplain data set (<https://github.com/hate-alert/HateXplain>)
- Built a model and training pipeline
- Classify input into one (or more) target groups out of 24
- Evaluation results:
 - F1: , 0.058, Precision: 0.3, Recall: 0.032
- Immense help from Fabian

I think that this text message is offensive. Please be nice and stick to the community guidelines.

Your hate was probably directed towards the following group(s): Women.

Meme Detection



- Contribution from Niklas
- We built the meme-detection-api
- In the bot the input image is classified: is it a meme?
 - If False: do nothing with the image
 - If True: hand the image over to the ocr-api and meme-model-api to classify

```
398 def detect_meme(url):
399     print("Start Meme Detection")
400     params = {"url": url}
401     r = requests.get(url=f"http://127.0.0.1:{PORTDICT['meme-detection-api']}/classifier", params=params)
402     is_meme = r.json()["result"]
403     print("is_meme: ", is_meme)
404     return is_meme
```


Documentation



- **Swagger**
 - Build with flask-apispec
 - Automatically builds Swagger documentation
 - Documentation can be used to test functionality
 - Later shown in demo
- **Readme**
 - Detailed instructions both for the end-user and anyone hosting an instance of the bot
 - Includes links to files that need to be downloaded

Deployment



- Didn't use Docker, couldn't get sudo rights on the uni server
- Instead:
 - Download the files as described in the README and move them to the corresponding folders
 - Run two scripts in the main folder:
 - Install.sh:
 - creates virtual environments and installs the requirements
 - Executes a command needed by the meme model
 - Run.sh
 - Fills a portdict file with an empty port per api
 - Starts each api in its own screen session, using the virtualenvs and ports
 - Starts the Modergator (telegram bot)
 - Prints all working screens
 - Prints the swagger-docu links for each API



Demo

Demo Plan

1. Im ersten Schritt: kurz **das Repo zeigen**, dass wir da die verschiedenen Ordner für die text apis angelegt haben, kurz beschreiben, dass man sich die beiden modell (target + meme-model) runter laden muss, weil die zu groß für git sind. Beschreiben, dass wir ohne Docker gearbeitet haben und dafür ein install.sh Skript haben, dass alle requirements installiert und zwei venvs anlegt. Eins für das meme model und eins für alle anderen, da das meme model eine ältere version von torch braucht. Das run.sh startet dann die apis jeweils in diesen virtualenvs in screens.
2. "Here you can see, that we have started all the apis and the bot in different screens. Since we have also created a **swagger documentation** for the apis, it prints the links to each documentation her, which we will show you soon."
Konsole: Ausführung von install.sh aus Zeitgründen überspringen (sollte davor aber einmal frisch mit vorher gelöschten venvs von der demo person ausgeführt werden.) Dann source run.sh ausführen. Wichtig: Nicht sofort Anfragen an den Bot schicken, noch ein paar Sekunden warten.
3. [wenn Docker geht: "We also build a docker compose file as Seid requested it. We did not have a Docker server for this project, so we realised this on one of our own pcs. To start docker, you have to...
4. **Swagger documentation:** text-api, demo with **"test text"** und **"you stupid fucking bot"**
5. Demo Bot in Telegram Web
 - a. **Text:** "his this is a test text" -> no result. Explain that we have added /debug for this reason. Activate /debug and send the message again.
 - b. **Offensive Text:** "you stupid fucking bot"
 - c. **Hate Text:** "you cunt"
 - d. **ASR normal:** send a voice message with "test test" , only shows debug message.
 - e. **ASR hateful:** Then send a voice message with "You fucking stupid bot idiot".
 - f. **Targets: Message offensive:** "All women are sluts and bitches hate them"
 - g. **Image** (not further processed): https://cdn.pixabay.com/photo/2019/03/14/23/17/rose-4056118_960_720.jpg

Demo Plan Teil 2

1. **Meme/OCR not hateful** : <https://blog.hslu.ch/diginect/files/2017/04/abbildung1-1024x538.jpeg>
2. -> and tell the group that it works with url AND uploaded images
3. **Meme/OCR hateful** url: <https://i.kym-cdn.com/photos/images/facebook/000/955/566/cb5.jpg>
4. Meme/OCR hateful image (vorher hier runterladen und per handy dann hochladen:
5. Meme/OCR hateful image with hateful text::
6. Meme/OCR not hateful image with hateful text:
7. Optin
8. Optout
9. Alle /help /start etc

Feedback Survey



<https://survey.lamapoll.de/Hate-speech-detection-Bot>

How do you rate the ...



Ease of use

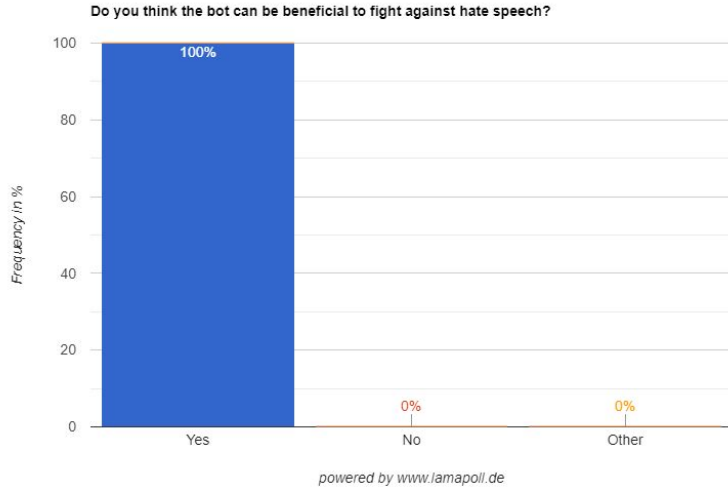
Explanations on how to use the bot

Performance of the bot

Other comments



Feedback Results



I like the pictures of the crocodile

There were many words that the bot did not consider hateful like []".

Answers for What do you think is the purpose of the bot?

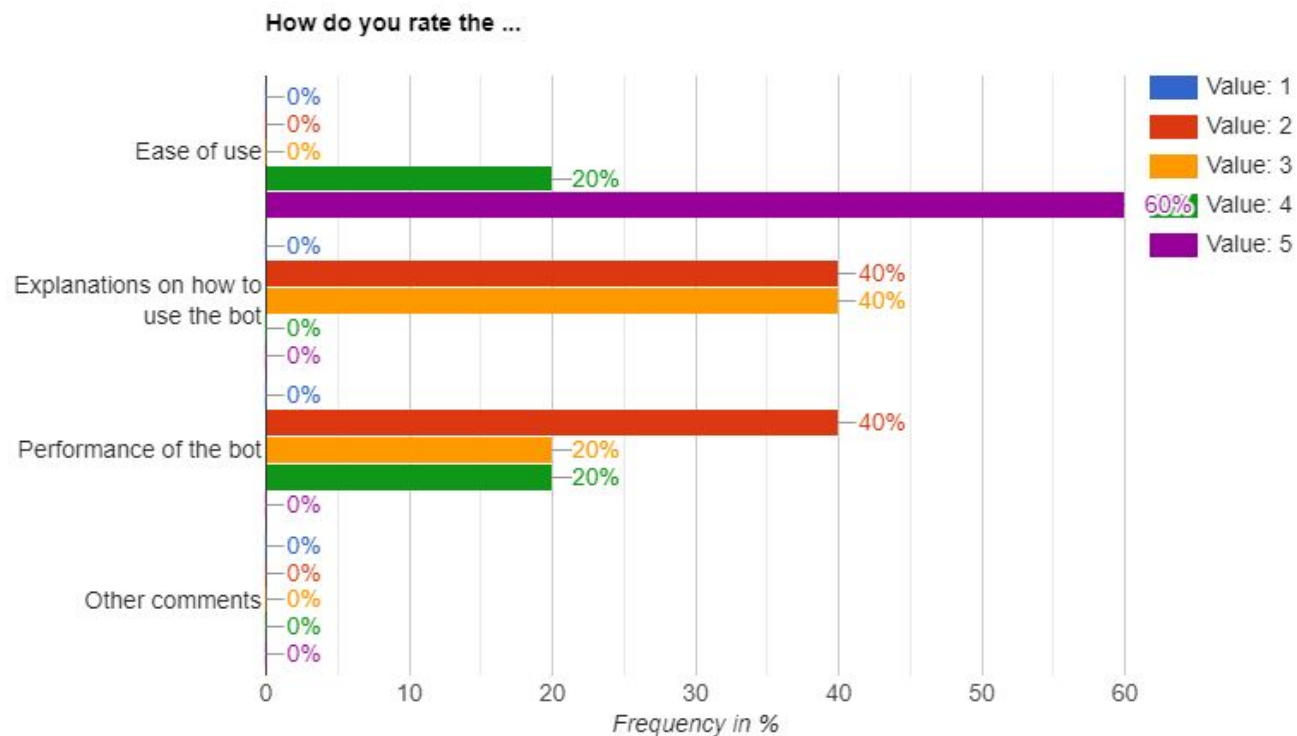
Detecting offensive and hate full messages

Remind people how they choose their words

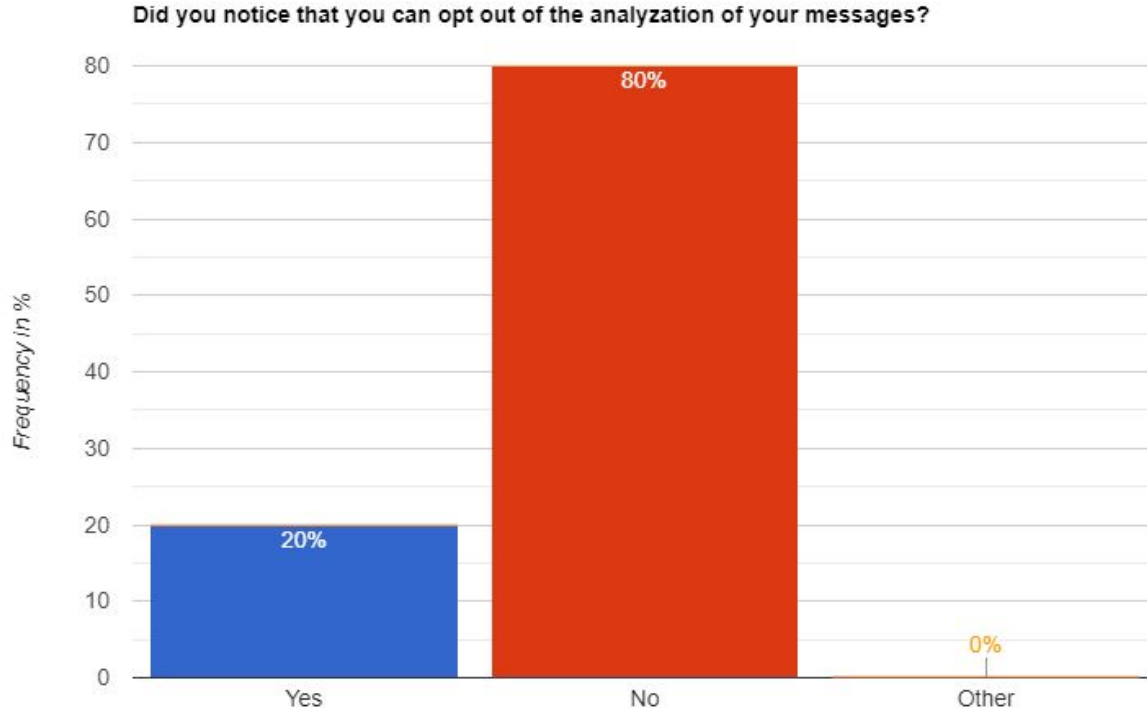
TO have an eye on groups to see if people behave and moderators are not fast enough

That people are nicer to each other

Feedback Results



Feedback Results



powered by www.lamapoll.de

Feedback (individual)



- The modergator was perceived as very cute and made the bot seem like a person
- Scores not clear
 - Introduced debugging mode and better explanation for the end-user ✓
- Unsure on how to use the bot (no reaction to messages)
 - Include /howto command to provide this information ✓

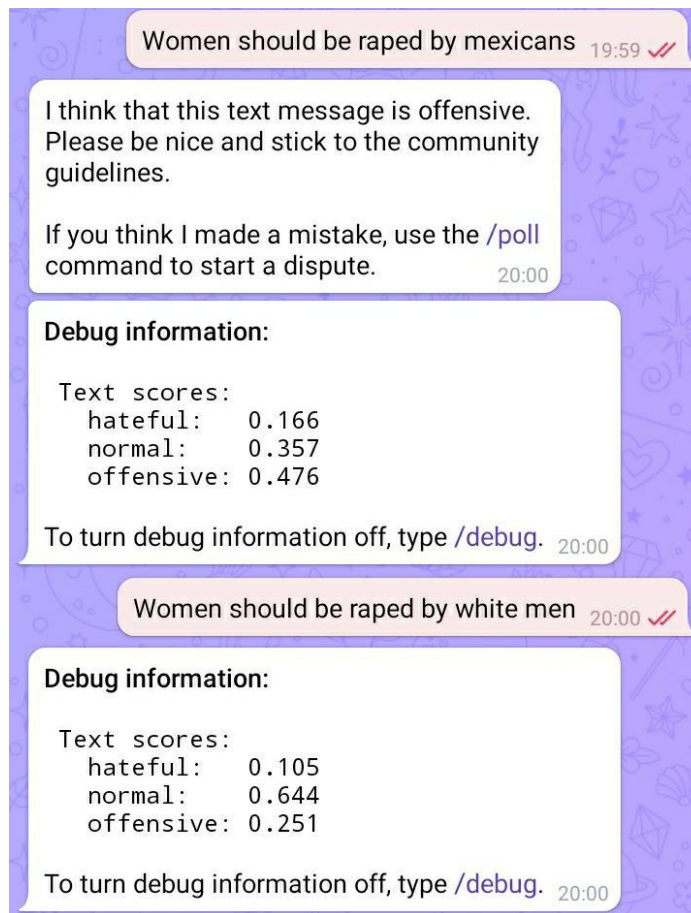
Feedback (individual)



- **Hateful messages are not always recognized as hate**
- Long image handling was unexpected
 - Add information to /start and /howto ✓
- "After a message is edited, there is no re-evaluation of the content"
 - Include an explanation in /howto ✓
 - Could be included in the future if possible
- Transcription text of the asr could also be beneficial for all users

Racial bias

- Probably due to data set or annotator's biases
- Other issue:
 - Detection of code words for discriminating people could be improved
 - African may be not the most adequate Name for a target group



The screenshot shows a chat interface with a purple background. At the top, a pink message bubble contains the text "Women should be raped by mexicans" with a timestamp of 19:59 and two red checkmarks. Below this, a white text box contains a report: "I think that this text message is offensive. Please be nice and stick to the community guidelines." followed by a suggestion: "If you think I made a mistake, use the /poll command to start a dispute." with a timestamp of 20:00. Below the report, a white box labeled "Debug information:" displays text scores: hateful: 0.166, normal: 0.357, and offensive: 0.476. At the bottom of this box, it says "To turn debug information off, type /debug." with a timestamp of 20:00. Below this, another pink message bubble contains the text "Women should be raped by white men" with a timestamp of 20:00 and two red checkmarks. Below this, another white box labeled "Debug information:" displays text scores: hateful: 0.105, normal: 0.644, and offensive: 0.251. At the bottom of this box, it says "To turn debug information off, type /debug." with a timestamp of 20:00.

Women should be raped by mexicans 19:59 ✓✓

I think that this text message is offensive. Please be nice and stick to the community guidelines.

If you think I made a mistake, use the /poll command to start a dispute. 20:00

Debug information:

Text scores:

- hateful: 0.166
- normal: 0.357
- offensive: 0.476

To turn debug information off, type /debug. 20:00

Women should be raped by white men 20:00 ✓✓

Debug information:

Text scores:

- hateful: 0.105
- normal: 0.644
- offensive: 0.251

To turn debug information off, type /debug. 20:00



Difficulties



- Build project without docker because of uni server restrictions
- Versioning of the different requirements
- Large files (too large for git)
- CPU vs GPU
- Adapting OCR code from Windows to Linux
- No time to do the evaluation of the meme-model
- Working on the uni servers

Progress



Plan

- Telegram Bot ✓
- Text Classification + API ✓
- Integration: OCR from Niklas ✓
- ASR Transcription + API (audio messages) ✓

After Midterm Stretch Goals:

- Prio 1: Target Classification + API ✓
- Prio 2 Feedback option for users in the bot
- ~~Prio 3: Store hate information for users.~~
- ~~Prio 4: Extend to Twitter, Discord or Slack as well~~
- ~~Evaluation of the Meme Model~~
- Bundle APIs and Documentation ✓

Add ons

- Modergator Design: Stickers, Slides, README etc. ✓
- Integration of Meme Detection from Niklas ✓
- Feedback study ✓
- ~~Integration of updated OCR from Niklas → aborted~~
- ~~Deployment with docker → aborted~~

Future work



- Include new ocr from Niklas with the confidence score
 - Was not possible due to technical challenges and time limitations
- Improve target detection
- Improve meme classification by training with transcription
- Make opt-out option more noticeable
- Retrieve more feedback to improve the user experience
- Collect feedback on the classification and improve the models
 - Lower threshold
- Extend to other social media platforms
- Extend to other languages (e.g. implement German or Persian from Imon)
- Logging (store hate information of users if compatible with DSGVO)

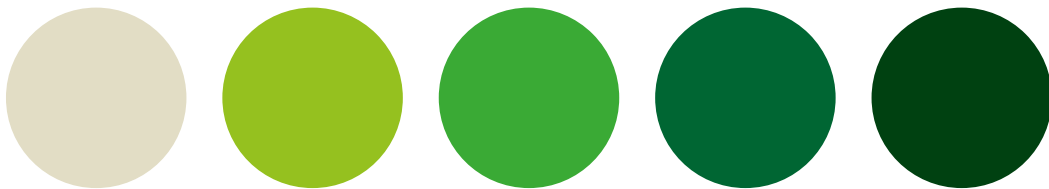
The End.



**Thank you for your
attention.**

Who Did What

Korbinian	Skadi	Katrin
Research telegram bot for groups + create prototype	Research + choose a meme model	
Design modergator logo & slides	Local prototype API and use HateXplain API	Made the meme model work
Modergator stickers	Research REST API with python	Meme model API
ASR api / Transcription	Target group classification	
run.sh & install.sh script	Interactive and informative bot: use polls, welcome messages, /help	(Small) test script to check APIs (inactive)
DSGVO/opt-out		Meme Detection API
Build APIs, connect modules with bot, (lots of) error fixing		Swagger API documentation
Detect images and URLs		Refactoring Telegram Bot
README	Target classification (build model, train, evaluate, connect to bot)	
Docker debugging	Retrieve and analyze feedback	
Create presentation, regular meetings, try demos, ...		



Fredoka One

Raleway

Timeline

May 6th - May 27th

- Research Telegram Bot for groups + create a prototype ✓
- Choose a meme model + research how to create the backend ✓
- Chose the second meme model to use for various reasons ✓
- Run the second meme model on a desktop pc ✓
- Decided on specs for a server and communicated with the LT group ✓
- Created a prototype API that runs locally ✓
- Researched and accessed the HateXplain API ✓
- Researched how to create the REST API with python (Flask) ✓
- Build a Telegram bot prototype and researched how to connect it with our API ✓

Timeline

May 28th - June 17th

- Build a dummy service and deploy on server ✓
- Take best model code + start creating backend + dummy API ✓
- Tried to run the model, predict a single image ✓
- Started working on the server basecpu1 ✓
- Used dockerized Flask API for scoring text input ✓
- Detect uploaded images and URLs ✓
- Use HateXplain locally and connect to the bot ✓
- Build Dummy OCR API to prepare integration from Niklas ✓

Timeline

June 18th - July 8th

Make models available via API and test functionality ✓

Integrate OCR*

Continuous documentation of components

Added image feature extraction to model pipeline ✓

Build Model API ✓

Decided on design of the bot, /goodvibes ✓

Began to include the target group ✓

Make bot interactive: included polls

Speech Recognition API ✓

Welcome message with DSGVO & Group info ✓

*from Niklas

Timeline

July 8th - July 25th

- exam period -

July 26th - Aug 19th

Include meme/non-meme classifier and OCR*

Bundle APIs ✓

Documentation of components ✓

Train model for target classification and create corresponding API ✓

Integrate target classification into Telegram Bot Messages via API ✓

Asked users for feedback ✓

Feedback option for users in the bot

Nice to have: Extend to other social media platforms

Aug 20th - Aug 26th

Finalize documentation and presentation ✓

*from Niklas