Application of Artificial Intelligence on online instant messaging platforms for the purpose of moderation

Author: Bojan Rađenović

The Twelfth Belgrade Grammar School and Regional Centre for Talented Youth Belgrade II, E-mail: bojan@radjenovic.dev
Supervisor: Mateja Opačić, *Regional Centre for Talented Youth Belgrade II*

1. Introduction

In today's society, most of our communication happens using the internet. For example, if we want to make plans with our friends or if we want to meet people with similar interests. However, I have noticed an issue with online chat rooms. The problem with public messaging platforms is bad behaviour and rudeness. The spread of negativity requires the presence of people to monitor the chat rooms, but this can be a demanding and sometimes unsuccessful process. I had experience with monitoring rooms and wanted something that could make the process easier. This project aims to facilitate the process of moderation by making a program that automatically analyzes messages and monitors chat rooms or forwards the messages for further analysis.

2. Material and Methods

The point of this project is to make a program based on artificial intelligence. The idea is to create a Discord Bot on the messaging platform Discord that would be able to perform automatic message analysis. A Discord Bot is a member of a chat room that is like a normal user. They are used to automate various actions using Discord's public API. Many Python libraries provide easier access to Discord's API. For message sentiment analysis, I decided to use Machine Learning. I used the TensorFlow library for model training and the disnake (1) library for interacting with the Discord API. I also have automated dataset collection. I created a Discord Bot, which automatically saves messages into a MySQL database from a public chat room. When I had collected around 30,000 messages, I decided to do pre-processing. This included deleting commas, periods, and some words. However, I also had to delete some messages from the database because they were repeated or in different languages. I ended up with a database containing about 20,000 messages. Due to the efficiency of training the model and the project itself, I used an already existing model from Google "Natural Language Processing Model" (2). This model from Google performs text embedding and is pre-trained using various news articles. I used my dataset with this model. Once I had finished training the model, I started working with the part that interacts with Discord. The disnake library works on the principle of events. For every action in the chat room itself, there is an event (user enters a room, exits, sends a message...). Using the "send message" event, I made the Discord Bot forward the message to the model for classification.

3. Results and Discussion

When I was working on the model itself, I had to change the parameters (learning rate, layers...) several times. Each training took about 10-15 minutes. I ended up with a model that was 80-90% accurate. Once I finished the Discord part, I decided to add the Discord Bot to multiple chat rooms. However, the Discord Bot would sometimes take the wrong action and disrupt conversations, so I decided to keep auto-monitoring as a side option. By default, the Discord Bot notifies the chat room staff instead of acting on its own.
Users also have the option to check if their message is classified as positive or negative by sending it privately to the Discord Bot.

4. Conclusion

For a project of minimal value, the idea was to create a program that would, on one platform, facilitate the monitoring of chat rooms. The program works exactly as intended. It assists in monitoring by forwarding questionable messages or by monitoring itself. In the future, I want to continue developing this project. The idea is to apply this model to other platforms that allow public chat rooms and to add support for multiple languages.

5. References

1. A Python library for interacting with the Discord API (https://github.com/DisnakeDev/disnake)
2. Google's model that does text wrapping (https://tfhub.dev/google/nnlm-en-dim50/2)