2023 IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI) | 979-8-3503-2110-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/SACI58269.2023.10158593

Automated Moderation Helper System Using Artificial Intelligence Based Text Classification and Recommender System Techniques

Barnabás Rőczey
John von Neumann Faculty of Informatics
Óbuda University
Budapest, Hungary
roczeybarnabas@stud.uni-obuda.hu

Sándor Szénási
John von Neumann Faculty of Informatics
Óbuda University
Budapest, Hungary
szenasi.sandor@nik.uni-obuda.hu

***Abstract*—In this paper we introduce a novel approach for text message moderation assistance, where the system can configure itself with negatively or positively labeled sample messages. The goal of the system is to assist moderators in moderating text messages, by providing them with recommendations about which messages should be reviewed manually. The recommendations should be based on user-defined rules, as well as machine-learned rules, derived from previously accepted or rejected messages. The proposed method is able to filter messages in Hungarian or English. It is also able to learn from user feedback, in order to better understand the preferences of users.**

***Index Terms*—natural language processing, machine learning, recommender system, text mining, microservice architecture, online moderation**

I. Introduction

Text message moderation is an essential part of many online services. It is used to ensure that messages posted by users adhere to the rules of the service and that they are appropriate for all users.

However, manual moderation of text messages can be time-consuming and tedious. It is also challenging to keep up with the sheer volume of messages that can be posted at any given time.

This paper proposes a novel system for text message moderation assistance. The system can be configured with sample messages labeled as either good or bad, along with the moderator that made that decision. It can then use those samples to learn how to identify messages that specific moderators would be interested in.

The system is designed to be modular and extensible, allowing for easy integration of new components, such as additional language support or a more advanced sentiment classification system. This makes it possible to adapt the system to changing needs quickly.

We evaluate the system using a dataset of real-world messages and benchmark the results against a baseline system.

II. Related Work

Text moderation has been a research topic for years, with many approaches being proposed. One common approach is to use a substring search. A more modern but expensive approach uses machine learning [1] to classify messages.

For example, researchers at Khulna University of Engineering & Technology has proposed a system for the automated detection of abusive text messages using a Naive Bayes classifier [2]. The system is trained using a dataset of annotated messages and can then be used to classify new messages.

Other systems, such as one developed by researchers at the Université de Lorraine, use deep learning to classify messages [3]. These systems are trained using a labeled message dataset and then used to classify new messages by that label.

The proposed system differs from these approaches in that it considers the differences between online communities and their moderators, allowing or disallowing certain behaviors based on the preferences of the moderators, even with no configuration. It is designed to be both modular and extensible. This allows the system to adapt to changing needs quickly and allows for easy integration of new components.

III. Methodology

A. The Architecture

In the proposed solution, we split the problem into multiple smaller problems and provide solutions using a micro-service architecture. Multiple of these systems are chained together to provide the final result. Some can be executed in parallel, while some depend on the output of one or more other components.

B. Flow of Information

First, we will describe the steps a message takes through processing. The intention is that the first processing methods in the chain will be less resource intensive and can provide a short-circuit trigger to stop processing early.

C. Text Analysis and Language Detection

The first two processing services can run in parallel. We generate basic text metrics such as unique 8-gram ratio, uppercase character ratio, lowercase character ratio, the ratio between uppercase and lowercase letters, and the ratio between letters and non-letter characters. Let us call this step text analysis. We can also run a language detection solution simultaneously, giving us the language code of the most likely language the message was written in.

D. More About the Text Analysis Step

The metrics provided by this are generally independent of language. The unique 8-gram ratio is derived by a sliding window taking 8 consecutive characters from the string at every position; then, we count how many of these are unique and calculate a ratio. We propose this is a simple yet effective method to find messages with repeated spam text.

A character is considered a letter if it has the Unicode character property of L (Letter); it is considered an uppercase letter if it has the Unicode character property Lu and lowercase if it has the property Ll. The letters of every language are classified in these categories, and some letters are neither uppercase nor lowercase.

The uppercase and lowercase ratio calculation is unique because this value can widely vary in case the lowercase count or the uppercase count is zero or one, while the other is a higher number. To ensure we get more valuable values, we take the base 10 logarithm of the ratio.

TABLE I
OUTPUTS OF THE TEXT ANALYSIS STEP

Input Text	**8-gr**	**upp**	**low**	**lett**	**up-lo**
”Lorem ipsum”	1	**0.09**	**0.82**	**0.91**	**-2.2**
”Lorem ipsum” repeated 2 times	**0.73**	0.09	0.82	0.91	-2.2
”Lorem imsum” repeated 3 times	**0.42**	0.09	0.82	0.91	-2.2
”Lorem IPSUM”	1	**0.55**	**0.36**	**0.91**	**0.41**
One sentence of lorem ipsum	1	0.02	0.82	0.84	-3.83
One sentence repeated 2 times	**0.53**	0.02	0.82	0.84	-3.83
One sentence repeated 3 times	**0.35**	0.02	0.82	0.84	-3.83
One paragraph of lorem ipsum	**0.94**	**0.02**	**0.81**	**0.83**	**-3.84**

As visible in Table I, we noted down the output values of specific input texts. ’8-gr’ stands for unique 8-gram ratio, ’upp’ represents the ratio of uppercase letters out of all characters, ’low’ represents the ratio of lowercase letters, ’lett’ represents the (Unicode) letter ratio, while ’up-lo’ is the uppercase to lowercase ratio.

E. More About the Language Detection Step

Certain character trigrams appear more often in some languages than others; we can grade the text based on this to detect the language. [4]

F. Language-Specific Profanity and Obscenity Detection

The next step in the processing chain is to run a language-specific profanity and obscenity detection algorithm. It utilizes both hash tables for fast lookups and regular-expression comparisons for certain swearwords that are often combined with other words or, in the case of agglutinative languages, conjugated words.

G. Short Circuit Option

At this point in the process, we have enough information to make decisions with accuracy comparable to mainstream methods; these metrics can be compared against the configured thresholds of each moderator, and in the case of a match, the message will be delivered to the moderator for review without further processing. This ensures no degradation of response time compared to current solutions.

H. Machine Translation Step

The step following is translation, this is only executed if the language is not English. After this step, all processing takes place on the translated message, but both the translated and the original text are sent to the moderator. We use the BART machine learning [5] translation model for translation. [6]

I. Lemmatization and Concept Gathering

After we ensured that we had the English version of the processed text, we ran lemmatization on it. Lemmatization is the process of converting a word to its base form by removing affixes. This is useful because words with the same meaning are grouped together, and we have less to classify. We reduce the number of lemmas we must classify by filtering out the stopwords. This step is directly followed by concept gathering, where certain lemmas have concepts linked to them, and those concepts can form a hierarchy with base concepts, such as the word ’Nazi’ has the concept ’politically incorrect’ and ’politically incorrect’ has the concept ’political’ as a linked concept. Certain words might be both sexual and profanity at the same time. This graph-based system allows us to set moderator-specific rules. One might want to ban politically incorrect themes, while another might want to ban all political conversations.

J. Lemma/Concept Token Interest Lookup and Prediction

The lemmatization and concept-gathering steps can be executed once per message, but the next step is moderator-specific. We predict an interest score between each lemma or concept and each moderator we wish to create recommendations for. The predictions are made the following way; first, we take training data as messages marked as essential and non-important by each moderator; we get the unique lemmas and concepts for each message and increase their relevance score of the matching concepts in the moderator’s profile if the message is flagged as necessary, if the message is flagged as unimportant, we increase an irrelevance score. This allows us to build a profile for each moderator and predict the relevance of each lemma or concept to that profile. The final lemma/concept interest score is calculated as follows; if the relevance score and the irrelevance score for a moderator-lemma pair add up to more than 10 (meaning that the moderator has rated messages with that lemma more than 10 times), then we calculate the ratio of the relevance score and irrelevance score and weigh it by the ratio of irrelevant messages to relevant messages, meaning that even if our data is unbalanced, the results will be balanced. If there are not enough samples to make a definite interest score for a lemma for a moderator, we predict the interest score as we do in a recommender system, using collaborative filtering. Namely, Bilateral Variational Autoencoder (BiVAE) based collaborative filtering is a generative model for dyadic data (e.g., user-item interactions). [7] When we calculate all lemma-based interest scores for a message, we take the maximum value as the interest score of the message.

We can also use this system to predict how interested the moderator is in a particular metric; for example, if a metric requires complex regular expressions or machine learning to measure, we can get a collaborative filtering score for each moderator and metric combination based on available data, then only run the specific analysis tool on the message if there are any moderators interested in it.

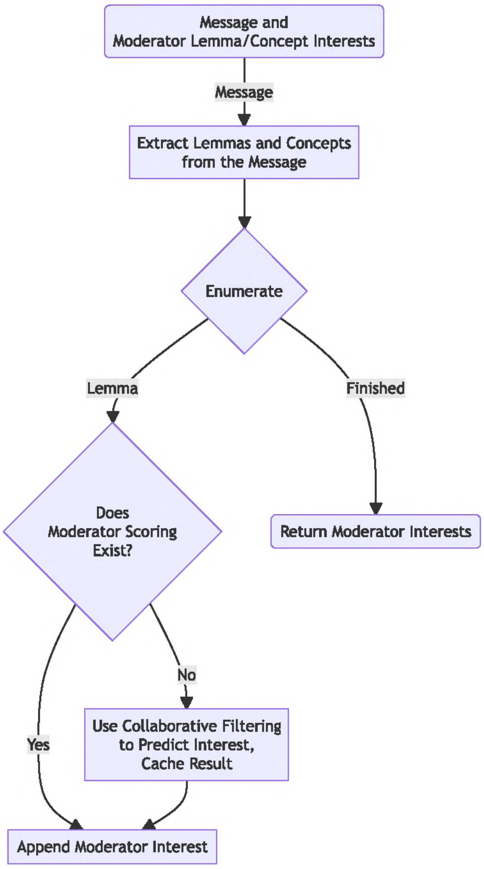
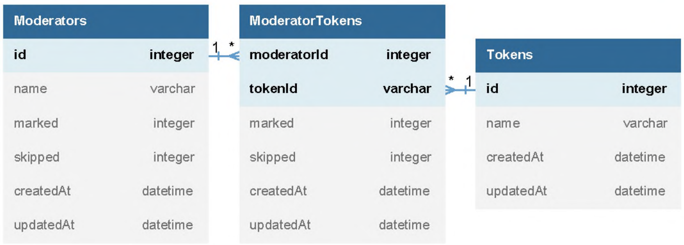

Message and
Moderator Lemma/Concept Interests
Message
Extract Lemmas and Concepts
from the Message
Enumerate
Lemma
Finished
Does
Moderator Scoring
Exist?
Return Moderator Interests
No
Yes
Use Collaborative Filtering
to Predict Interest,
Cache Result
Append Moderator Interest

Fig. 1. The lemma and concept based interest calculation.

As visible in Figure 1, the input to this subsystem is the message we wish to evaluate and a database containing the calculated interest scores between moderators and tokens (lemmas or concepts). We can use this database for lookups as well as to do collaborative filtering.

The output is an interest score for each token in the message.

Figure 2 shows that this normalized database scheme counts messages that the moderator rates and interest (marked) or non-interest (skipped) counts for tokens.



Moderators	
id	integer
name	varchar
marked	integer
skipped	integer
createdAt	datetime
updatedAt	datetime

ModeratorTokens	
moderatorId	integer
tokenId	varchar
marked	integer
skipped	integer
createdAt	datetime
updatedAt	datetime

Tokens	
id	integer
name	varchar
createdAt	datetime
updatedAt	datetime

Fig. 2. Lemma/Concept database structure.

We also can use the createdAt and updatedAt fields to remove values that have not been seen in a long time, such as tokens with randomly generated URLs or randomly pressed keys.

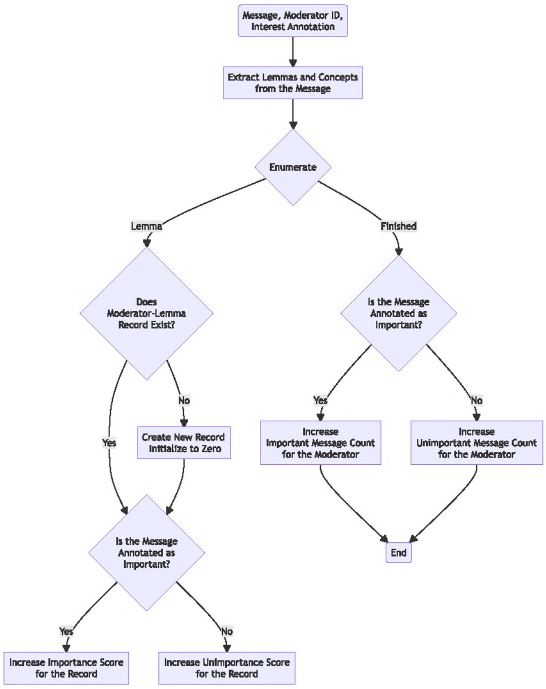


Fig. 3. Analysis of labeled messages.

Figure 3 shows the process used to gather data about the moderator and token relations for the database mentioned above.

K. Moderator-Profile Based Zero-Shot Machine Learning

Finally, we process the message using a more resource-intensive machine learning approach. We have a neural network consisting of text embedding layers, profile encoding layers, and a profile-based recommender system that uses the encoded profile to decide an interest score for the encoded message. The profile consists of the outputs of the text embedding layers with previously classified messages and the manually set interest rating.

Figure 4 shows the structure of our complete neural network. However, not all parts are used for every step during

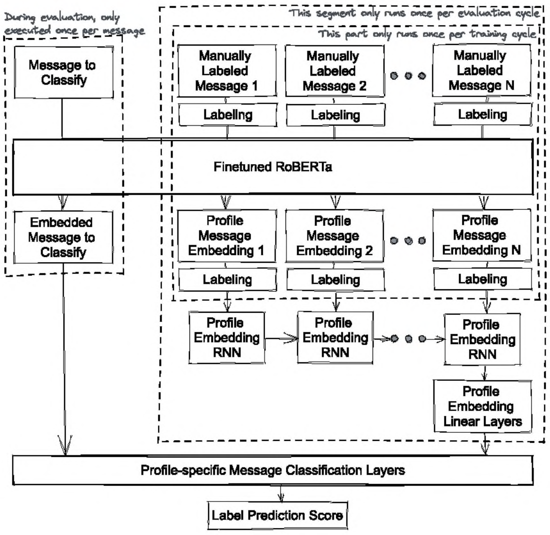

During evaluation, only executed once per message
This segment only runs once per evaluation cycle
This part only runs once per training cycle
Message to Classify
Manually Labeled Message 1
Manually Labeled Message 2
Manually Labeled Message N
Labeling
Labeling
Labeling
Finetuned RoBERTa
Embedded Message to Classify
Profile Message Embedding 1
Profile Message Embedding 2
Profile Message Embedding N
Labeling
Labeling
Labeling
Profile Embedding RNN
Profile Embedding RNN
Profile Embedding RNN
Profile Embedding Linear Layers
Profile-specific Message Classification Layers
Label Prediction Score

Fig. 4. The structure of our neural network.

training or evaluation, and the outputs of specific steps can be re-used.

The labeling is a property of the messages that make up the moderator's profile; however, they are not used during the RoBERTa step; instead, they are attached to the output of that step and provided as input to the profile embedding RNN.

The labeling stands for whether the message was marked as interesting by the moderator or not.

Figure 5 visualizes the complete message analysis chain in our system.

'Lemmatization and Concept Gathering With Moderator Interest' stands for Fig. 3.

'Message Embedding' and 'Classification Using the Moderator's Profile' represent evaluation steps shown in Fig. 4.

IV. Evaluation

In order to evaluate the system, we will use a dataset of real-world messages and benchmark the results against a baseline system.

Our dataset is already automatically filtered by a baseline system; thus, any observation our proposed system makes is something the baseline system missed.

The dataset contains messages that moderators have manually removed, messages that have not been removed, and messages that have been automatically removed by the baseline moderation system but manually approved by a moderator.

Our training data is gathered from the site called Reddit, and the Moderator column represents a community on the site with different rules, the column titled Normal represents the count of messages in the dataset that have not been marked as necessary, the Removed column contains the count of messages manually removed by the moderator. In contrast, the Approved column has the number of messages in the dataset removed by the baseline moderation system, but later the moderator approved (Table II).

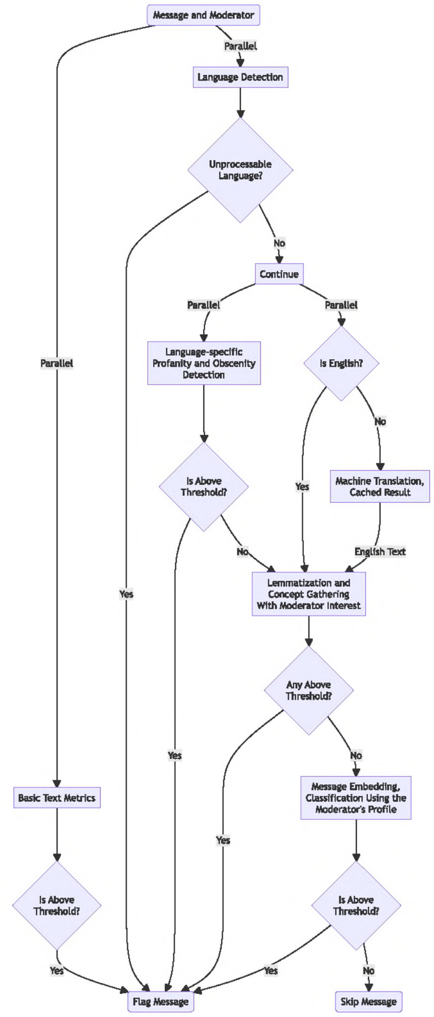


Fig. 5. The message processing flow.

TABLE II
Sample of Training Data Statistics

Moderator	Normal	Removed	Approved
AskReddit	135024	134304	6003
teenagers	118995	115789	2590
CryptoCurrency	38371	37060	546
memes	33882	33748	1560
NFTsMarketplace	35064	33473	89
news	33285	32081	407
AmItheAsshole	26104	26864	2115
nfl	20184	19394	191
worldnews	17663	18240	1496
selfie	18651	17907	157
soccer	17958	17721	657
antiwork	17445	17253	679
relationship_advice	16919	16870	801
AskMen	15753	15525	554
Tinder	15530	15031	260
politics	14023	14034	723
shitposting	13866	13966	808
amihot	13959	13358	74
nba	13191	12949	414
SquaredCircle	13328	12817	256
...	...	...	...
SUM	1726445	1702722	64943

TABLE III
SAMPLE OF THE INFERRED TOKEN INTERESTS

Moderator	**Token**	**Interest**	**Explanation**
amihot	pz2mnldleehjcj32g6	463	The token is a link to an undesirable animated GIF.
Christianity	U+200B	315	A zero-width space, either used to avoid substring search or it can appear in text copied from documents.
Christianity	37:11	126	A Bible quote about the land belonging to the believers.
Wallstreetsilver	oligarch	112	This phrase often appears in conspiracy theories.
wallstreetbets	grain	63	The concept that grain is the best investment due to economic collapse.
leagueoflegends	intensive	57	A joke about which playable characters are difficult to play.
conspiracy	lamestream	53	Derogatory term for mainstream media.
ukraine	Mordor	52	A term to dehumanize Russians, Russians are orcs from Mordor.
meirl	instagram.com/...	48	Instagram advertisement.
Minecraft	Minetest	41	A game that's competition to Minecraft.
AMA	DV	40	Diversity Visa lottery, US immigration by luck.
news	jure	33	”de jure”, often used to refer to a legal transition to a system considered illegal, such as de jure fascism, de jure apartheid.
news	punchable	27	”punchable face” targeted insult against a person.
ukraine	galactic	25	”Galactic Empire”, a Star Wars reference to the dark side, used to refer to the Russians.
IdiotsInCars	CVC	25	Quoting the California Vehicle Code in arguments.
antiwork	indoctrination	24	The concept that people that don't question why they work have been indoctrinated.

In Table III, we provide moderators, tokens, and the calculated interest for some example phrases that the baseline system missed but our system could detect.

We also explain why those phrases are controversial in their specific communities.

It is worth noting that even though the system scored the interest of moderators in concepts such as politics, and obscenity, specific terms could achieve even higher scores. Also, obscene communities and phrases have been omitted from the example table.

V. CONCLUSION

In this paper, we proposed a novel system for text message moderation assistance. The system can be configured with sample messages that are labeled as either good or bad, along with the moderator that made that decision. Using this information, the system can then learn from the preferences of the moderators and provide them with recommendations about which messages should be reviewed manually.

The system is designed to be both modular and extensible, allowing for easy integration of new components, such as additional language support or a more advanced sentiment classification system. This makes it possible to adapt the system to changing needs quickly.

We evaluated the system using a dataset of real-world messages and benchmarked the results against a baseline system. The results show that our system can identify messages that the baseline system missed and that it can be used to assist moderators in moderating text messages.

Both the lemma and concept-based recommendation systems and the profile-based recommendation systems can learn from the output. If a user accepts or rejects a suggestion, we can update the model to reflect the user's preferences better.

This system is designed to be both modular and extensible, allowing for easy integration of new components, such as additional language support or a more advanced concept-gathering system. This makes it possible to adapt the system to changing needs quickly.

ACKNOWLEDGMENT

The authors would like to thank the High Performance Computing Research Group of Óbuda University for its valuable support. The authors would like to thank NVIDIA Corporation for providing graphics hardware for the experiments.

REFERENCES

[1] M. Nevendra and P. Singh, “Software defect prediction using deep learning,” *Acta Polytechnica Hungarica*, vol. 18, no. 10, pp. 173–189, 2021.

[2] M. A. Awal, M. S. Rahman, and J. Rabbi, “Detecting abusive comments in discussion threads using naïve bayes,” in *2018 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, 2018, pp. 163–167.

[3] A. G. D'Sa, I. Illina, and D. Fohr, “Towards non-toxic landscapes: Automatic toxic comment detection using DNN,” *CoRR*, vol. abs/1911.08395, 2019. [Online]. Available: http://arxiv.org/abs/1911.08395

[4] G. Grefenstette, “Comparing two language identification schemes,” in *Proceedings of JADT*, vol. 95, 1995.

[5] A. Pejić and P. S. Molcer, “Predictive machine learning approach for complex problem solving process data mining,” *Acta Polytechnica Hungarica*, vol. 18, no. 1, pp. 45–63, 2021.

[6] Yang Zijian Győző, “BARTerezzünk! - Messze, messze, messze a világtól, - BART kísérleti modellek magyar nyelvre,” in *XVIII. Magyar Számítógépes Nyelvészeti Konferencia*. Szeged, Magyarország: Szegedi Tudományegyetem, Informatikai Intézet, 2022, pp. 15–29.

[7] Q.-T. Truong, A. Salah, and H. W. Lauw, “Bilateral variational autoencoder for collaborative filtering,” in *ACM International Conference on Web Search and Data Mining, WSDM 2021*, 2021.