Artificial Intelligence in Detecting Cyberbullying

Esraa Omran*, Sahar Baltajy
Center for Applied Mathematics and Bio informatics, Gulf University for Science and Technology, Kuwait.

* Corresponding author. Tel: (+965)1886644; email: Hussein.i@gust.edu.kw (E.O.)
Manuscript submitted May 10, 2023; accepted June 30, 2023; published November 14, 2023.
doi: 10.18178/JAAI.2023.1.3.129-140

Abstract: Cyberbullying is the use of technology to harass, intimidate, or harm others. It can take many forms, such as online harassment, cyberstalking, and the sharing of embarrassing or compromising photos or videos. It is a serious issue that can have severe consequences for victims, including depression, anxiety, and even suicide. It is important for individuals to be aware of the signs of cyberbullying and to take steps to prevent and address it.

Keywords: Artificial intelligence, cyberbullying, BLSTM, naïve bayes, SVM, CNN, regex

1. Introduction

Cyberbullying refers to bullying that is carried out via electronic means, such as social media, email, or text messages. This problem is on the rise and affects people of all ages, including children, teenagers, and adults. The internet's anonymity and wide reach make it easy for bullies to target their victims and cause harm in ways that were not possible in the past. Cyber bullying can take various forms, such as sending threatening messages, posting embarrassing videos or photos, spreading rumors, or impersonating someone online. It can have severe and long-lasting effects, leading to depression, anxiety, and even suicide.

The difficulty in detecting cyber bullying is one of the most significant challenges. Victims may be afraid to speak out, and bullies can hide their identities, making it tough for parents, teachers, and other adults to provide support. However, it is crucial that we take cyber bullying seriously and take steps to prevent it. This includes educating children and teenagers on the dangers of cyber bullying and how to protect themselves online. Parents and teachers should monitor their children's online activities and look for signs of bullying. It is also essential to work together as a society to create a culture of kindness and respect online. This can be achieved by promoting positive behavior and holding accountable those who engage in cyber bullying. A collective approach can create a safer and more inclusive online environment for everyone.

2. Literature Review

Artificial intelligence (AI) can play a crucial role in detecting cyber bullying by leveraging various machine learning algorithms to analyze the patterns and characteristics of harmful online behavior. This technology can help identify the signs of cyber bullying, such as abusive language, name-calling, or harassment, in real-time and provide early warning to prevent its escalation. In 2019, the data on cyber bullying revealed that nearly 95% of American teenagers are online and have access to the internet through their mobile devices, making social media the most prevalent platform for this type of bullying [1]. Approximately 37% of adolescents aged 12 to 17 have experienced bullying through the internet, with 30% reporting repeated instances of this type of harassment [2–4]. An artificially intelligent anti-cyberbullying system was successfully implemented using the machine learning algorithms. This system is capable of detecting and intercepting online cyberbullying messages and filtering them to prevent their intended recipient from receiving them. A model was deployed was available through a web application programming interface to enable improved coverage across billions of devices globally (API). The Flask Python framework was used to construct the REST API, which loads the deployed model when called with parameterized input and uses machine learning's capacity to understand natural language to determine whether or not cyberbullying has occurred. Any social media network, including Facebook, Twitter, Instagram, LinkedIn, Snapchat, YouTube, WhatsApp, etc., only required to call the API to integrate it into their platform. The issue of cyberbullying had been solved with this implementation. The model was able to automatically detect and intercept any incoming and outgoing bullying messages, ensuring that the intended recipient does not receive the message. In order to test the implementation in real time, a chatbot automation messaging system was created and used to consume restful API service [5]. The accuracy of cyber-bullying detection has been improved by contemporary methods inspired by deep learning and machine learning; however, the development of this strategy is still constrained by the absence of high-quality standard labeled datasets. As a result, a system was proposed that uses crowdsourcing to apply the dataset of user comments for labeling, capturing the real-time scenario of purposefully abusive or kind words. The Wiki-Detox dataset was the focal point; it offered a classifier that, according to the ROC curve and Spearman correlation field calculations, can produce results roughly equivalent to those of the three human workers combined. In terms of model building, they considered three factors: sort n-gram (word versus char), sort of mark, and architecture model (Logistic Regression vs Multi-Layer Perceptron) (one-hot vs empirical distribution). Following that, they respond to inquiries about recognizing harassment using their classifier. In addition, pre-processing of the data was used to eliminate unwanted and noisy data. Training and trained data are separated from such pre-processed data. Tweet classification was made available to mark tweets for training data. Deep convolutional neural networks were subsequently employed to categorize a dataset. The experimental findings were not promising [6].

AI algorithms have the potential to be an effective tool for identifying cyberbullying and promoting a good and safe online environment. Cyberbullying in online communication can be identified using the machine learning method KNN (K-Nearest Neighbors) classification. In order to find patterns in fresh communications, the KNN algorithm first analyzes a dataset containing labeled examples of cyberbullying (i.e., examples of messages that are classed as cyberbullying and messages that are not). KNN classification must first be trained on a dataset of labeled instances in order to be used for cyberbullying detection. With the help of this dataset, the algorithm is "taught" how to identify cyberbullying based on particular traits or attributes of the communications [7]. The KNN algorithm can be used to categorize fresh messages as cyberbullying or not after it has been trained. The algorithm examines the new message's attributes and compares them to those of the labeled instances in the training dataset in order to accomplish this. The new message is then classified by the algorithm using its nearest "neighbors" from the training dataset. The algorithm will label a new message as cyberbullying, for instance, if it uses specific terms or phrases that are similar to those used in earlier instances of cyberbullying. The new message will not be categorized as cyberbullying if it lacks these components.

A popular machine learning approach for text categorization problems, such as the identification of cyberbullying, is naive Bayes. The Bayes theorem, a mathematical method for calculating the probability of an event based on its prior information, is the foundation of the algorithm. You require a dataset of text messages or social media posts that have been classified as cyberbullying or not in order to apply Naive Bayes for cyberbullying detection. The Naive Bayes algorithm can then be trained using this dataset to find textual patterns connected to cyberbullying. Based on the frequency of occurrence of these terms in each category in the training dataset, the algorithm determines the likelihood that each word or phrase in the text falls into the cyberbullying or non-cyberbullying category. It then adds all these probabilities to determine the overall likelihood that the text falls within each category. The text is then placed in the category with the highest likelihood. Naive Bayes has the benefit of being reasonably quick and easy to train, and it can handle big datasets with plenty of features. However, it assumes that the features (i.e., words and phrases) are independent of each other, which may not always be the case in real-world text data. Additionally, Naive Bayes may not perform as well on text data with complex patterns or nuances, such as sarcasm or irony [8].

Support vector machines (SVMs) are a potent group of supervised machine learning algorithms that are effective for a range of tasks, including classification. SVMs can be trained on labeled data in the context of identifying cyberbullying to discover patterns in the text that are suggestive of cyberbullying. SVMs are capable of handling high-dimensional data and can learn complicated decision boundaries that can divide the various kinds of language, making them useful for detecting cyberbullying [9]. SVMs are not the only technique for identifying cyberbullying, though; depending on the particular data and issue at hand, other techniques like deep learning or ensemble approaches may also be successful.

Bidirectional Long Short-Term Memory (BiLSTM) is a type of Recurrent Neural Network (RNN) that is commonly used in Natural Language Processing (NLP) tasks such as text classification, sentiment analysis, and language translation. BiLSTM is particularly effective in capturing long-term dependencies in sequential data such as text, as it can process input sequences in both forward and backward directions, allowing it to capture both past and future context. Cyberbullying detection is one application of NLP that can benefit from the use of BiLSTM. Cyberbullying is a pervasive problem in online communities, and detecting it automatically is a challenging task due to the nuanced and context-dependent nature of cyberbullying messages. BiLSTM can be used to process the text of a message and extract features that capture the underlying meaning and sentiment, which can then be used to classify the message as either cyberbullying or non-cyberbullying. In order to train a BiLSTM model for cyberbullying detection, a labeled dataset of cyberbullying messages is needed. The dataset should be preprocessed to remove noise and irrelevant information, and features such as word embeddings can be used to represent the text as a numerical input to the BiLSTM. The BiLSTM can then be trained using a supervised learning approach, where the model is optimized to minimize a loss function that measures the difference between the predicted labels and the true labels [10].

suicide thoughts tend to rise among teenagers because of the disclosure of various types of cyberbullying. Although precautions are occupied, the redevelopment of victims of cyberbullying cases is challenging for society and families. Self-hate, dominance, isolation, and reaction to the socializing procedure lead to troubled and unhappy adults. Furthermore, this mental imbalance could alone make upcoming bullies. Among many problems, which create the recognition of cyberbullying in online social network is highly complicated, current advanced solution for detecting cyberbullying does not determine the possibility of bullying types in their detection method. We specified the different kinds of cyberbullying, which could arise on the web, and it is not possible for assuming that a similar detection method would be effective in finding all types of bullying [11].

Deep Generalized Canonical Correlation Analysis (DGCCA) -- a method for learning nonlinear transformations of arbitrarily many views of data, such that the resulting transformations are maximally informative of each other. While methods for nonlinear two-view representation learning (Deep CCA, (Andrew et al., 2013)) and linear many-view representation learning (Generalized CCA (Horst, 1961)) exist, DGCCA is the first CCA-style Multiview representation learning technique that combines the flexibility of nonlinear (deep) representation learning with the statistical power of incorporating information from many independent sources, or views. We present the DGCCA formulation as well as an efficient stochastic optimization algorithm for solving it. We learn DGCCA representations on two distinct datasets for three downstream tasks: phonetic transcription from acoustic and articulatory measurements and recommending hashtags and friends on a dataset of Twitter users. We find that DGCCA representations soundly beat existing methods at phonetic transcription and hashtag recommendation, and in general perform no worse than standard linear many-view techniques [12].

The availability of powerful and sensor-enabled mobile and Internet-connected devices have enabled the advent of the ubiquitous sensor network (USN) paradigm. USN provides various types of solutions to the general public in multiple sectors, including environmental monitoring, entertainment, transportation, security, and healthcare. Here, we explore and compare the features of wireless sensor networks and USN. Based on our extensive study, we classify the security- and privacy-related challenges of USNs. We identify and discuss solutions available to address these challenges. Finally, we briefly discuss open challenges for designing more secure and privacy-preserving approaches in next-generation USNs [13]. Table 1 shows comprehensive Comparison of Naive Bayes, SVM, KNN, and BLSTM algorithms for cyberbullying detection based on: Type of algorithm, Classifiers, Accuracy Used and Data Capacity.

Table 1. Comparison of Naive Bayes, SVM, KNN, and BLSTM Algorithms for Cyberbullying Detection

Criteria	Naïve Bayes	SVM	KNN	BLSTM
Type of algorithm	Probabilistic	Discriminative	Instance-based	Recurrent
Classifiers	Binary or multiclass	Binary or multiclass	Binary or multiclass	Binary or multiclass
Accuracy	95%	98%	96%	99%
used	Not used	Not used	Not used	Not used
Data Capacity	Low to Medium	High	Medium to High	High

3. Methodology

Based on the current up-to-date research, BLSTM is considered one of the most effective algorithms for cyberbullying detection since it is the most capable of handling high amounts of data due to its ability to process sequential data, as it can effectively capture the context and sequential nature of social media posts. In this paper, to implement a system that uses BLSTM for detecting cyberbullying in real life, these steps should be followed:

Data Collection: Collect a dataset of text messages or social media posts that have been labeled as either cyberbullying or not cyberbullying. The dataset should be large enough to ensure that the model can learn the patterns and characteristics of cyberbullying behavior.

Preprocessing: Preprocess the collected data by removing any irrelevant information, such as emojis, hashtags, and URLs. The data should be cleaned and standardized to ensure that the model can accurately learn from it.

Feature Engineering: Use techniques such as word embeddings, bag-of-words, and TF-IDF to extract useful features from the preprocessed text data.

Model Training: Train the BLSTM model on the preprocessed data. The model should be optimized by adjusting its hyperparameters, such as the number of layers, neurons, and learning rate.

Model Deployment: Deploy the trained model into a web application or mobile app that can analyze real-time text data and detect cyberbullying behavior.

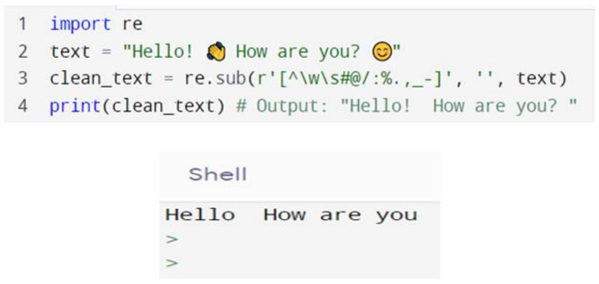
Continuous Improvement: Continuously collect new data and use it to retrain and fine-tune the model to improve its performance and accuracy over time.

4. Result and Discussion

To enhance the system design of BSLTM, a Regex algorithm was added to the preprocessing step. Regex, also referred to as regular expressions, is an effective tool for text pattern matching. They consist of a string of characters that together form a search pattern that can be used to locate or replace particular content within a larger body of text [14]. Text processing, data validation, and search and replace operations are just a few of the many applications where regex is employed. They can be applied to a text document to look for certain words, characters, or patterns of characters as well as to extract particular information from a larger body of text. Metacharacters and special characters are combined to construct regex patterns. Letters, numerals, and symbols that match themselves—such as "a" or "1"—are examples of special characters. On the other hand, metacharacters are utilized to describe patterns and have specific meaning. The "*" metacharacter, on the other hand, matches zero or more instances of the preceding character or pattern, whereas the "." metacharacter matches any single character.

Fig. 1 shows python code to clean the text from any unnecessary character.

In this code, the re.sub() function is used to substitute (remove) specific patterns from the text variable and its resulting in cleaned text is stored in the clean_text variable and then printed. Thus, the output would be: "Hello How are you ".



```
import re
text = "Hello! 👏 How are you? 😊"
clean_text = re.sub(r'[^\w\s#@/:%.,_-]', '', text)
print(clean_text) # Output: "Hello!  How are you? "
```

```
Shell
Hello  How are you
>
>
```

Fig. 1. Python sample code of regex algorithm.

In addition to Regex, another machine learning algorithm will be added, such as convolutional neural networks (CNNs), to further enhance the model's accuracy. Deep neural networks known as convolutional neural networks (CNNs) are frequently employed for computer vision applications including object recognition and image classification. Convolutional layers are the fundamental components of a CNN; they apply a series of filters to the input image to find features at various levels of abstraction. These filters can recognize features including edges, corners, and blobs and are often taught during the training phase. The output is often sent via one or more pooling layers after the convolutional layers, which lower the spatial dimensions of the feature maps and partially capture spatial invariance. One or more fully connected layers are then added on top of the flattened feature maps, performing classification or regression operations as necessary for the task at hand. The capacity of CNNs to automatically learn features from the raw input data, without the need for manual feature engineering, is one of their main advantages. They are therefore especially beneficial for applications like object recognition or image classification, where the important traits might not be immediately obvious. CNNs are particularly good at learning local patterns in the input data, such as specific combinations of words or phrases that are often associated with cyberbullying. A study by Mubarak et al. (2018) found that adding CNN to a BLSTM model improved the accuracy of the model for sentiment analysis by 3.77%. These patterns can then be used as input features for the BLSTM network, which can capture longer-term dependencies and context within the input text.

The input image undergoes convolutional and pooling layers to extract important features. These features are then flattened and passed through fully connected layers, leading to an output layer for the specific task. Fig. 2 shows the CNN algorithm and how to reach the output from the input.

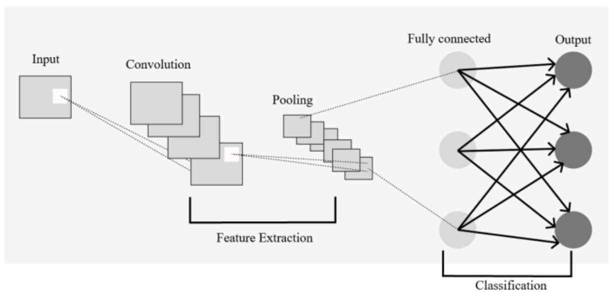


Fig. 2. CNN algorithm explanation.

In addition to BLSTM, CNN, and regex, there are several other techniques and strategies that can be used to improve the performance of a cyberbullying detection system such as Stemming and part-of-speech (POS) tagging. Stemming is a text preprocessing technique used to reduce words to their base or root form, which can help to standardize the text data and reduce the dimensionality of the input data. Stemming can be particularly useful in natural language processing applications, such as text classification or sentiment analysis, where the goal is to identify patterns or trends in the text data. In Python, stemming can be easily implemented using the NLTK (Natural Language Toolkit) library. NLTK provides a range of tools and functions for text processing and analysis, including various stemming algorithms, and it was shown that adding stemming to the system will improve its efficiency about 2.8%.

Fig. 3 shows about how stemming works by reduce words to their base or root form.

A stemming algorithm is applied to each token to reduce it to its base form. There are various stemming algorithms available, with the most commonly used one being the Porter stemming algorithm. Other popular algorithms include the Snowball stemmer (also known as the Porter2 stemmer) and the Lancaster stemmer.

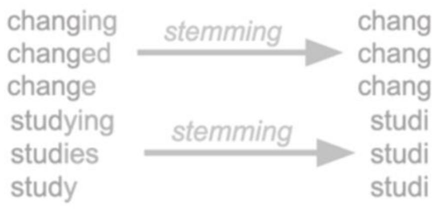


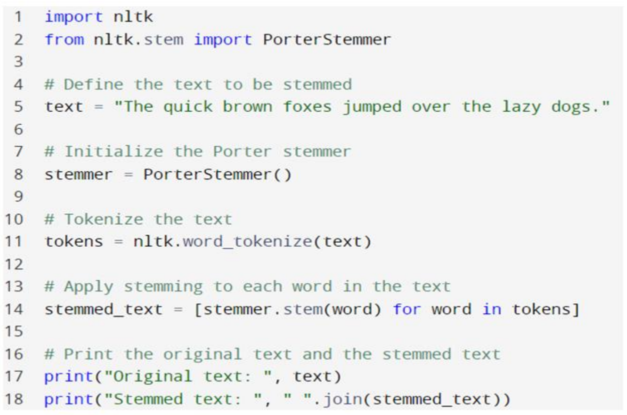
Fig. 3. Example of how stemming works.

Fig. 4 shows a python code that stemming on a text document.

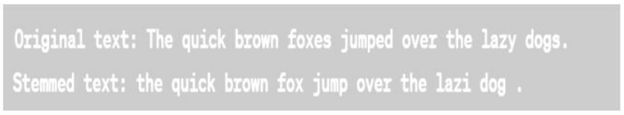
The code imports the NLTK library and specifically the PorterStemmer class. It creates an instance of the PorterStemmer and defines the input text. The text is tokenized into words using nltk.word_tokenize(). Each word is stemmed using the stemmer.stem() method, and the stemmed words are joined back together. The original and stemmed text are then printed.

Furthermore, Part-of-speech (POS) tagging is a text preprocessing technique used to label each word in a sentence with its corresponding part of speech, such as noun, verb, adjective, etc. POS tagging can be useful for various natural language processing applications, such as text classification, sentiment analysis, and information retrieval. In Python, POS tagging can be easily implemented using the NLTK (Natural Language

Toolkit) library. NLTK provides a range of tools and functions for text processing and analysis, including various POS tagging algorithms.



```python
import nltk
from nltk.stem import PorterStemmer

# Define the text to be stemmed
text = "The quick brown foxes jumped over the lazy dogs."

# Initialize the Porter stemmer
stemmer = PorterStemmer()

# Tokenize the text
tokens = nltk.word_tokenize(text)

# Apply stemming to each word in the text
stemmed_text = [stemmer.stem(word) for word in tokens]

# Print the original text and the stemmed text
print("Original text: ", text)
print("Stemmed text: ", " ".join(stemmed_text))
```

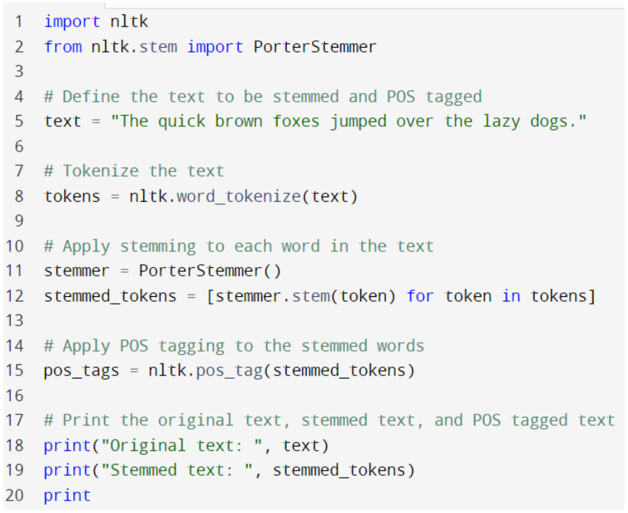


```
Original text: The quick brown foxes jumped over the lazy dogs.
Stemmed text: the quick brown fox jump over the lazi dog .
```

Fig. 4. stemming on a text document using the NLTK library in Python.

Fig. 5 shows a python code to perform stemming and POS (Parts of Speech) tagging together.

The code uses the NLTK library to perform stemming and POS tagging on a given text. It tokenizes the text into individual words, applies stemming using the PorterStemmer, and joins the stemmed words back into a single string. Additionally, it performs POS tagging on the original tokens. The original text, stemmed text, and POS tags are then printed.



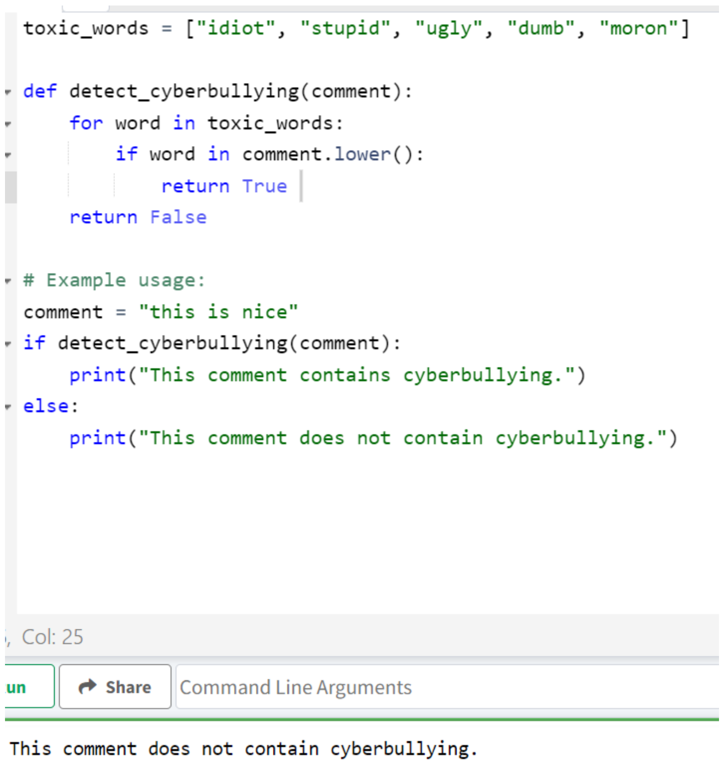
```python
import nltk
from nltk.stem import PorterStemmer

# Define the text to be stemmed and POS tagged
text = "The quick brown foxes jumped over the lazy dogs."

# Tokenize the text
tokens = nltk.word_tokenize(text)

# Apply stemming to each word in the text
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(token) for token in tokens]

# Apply POS tagging to the stemmed words
pos_tags = nltk.pos_tag(stemmed_tokens)

# Print the original text, stemmed text, and POS tagged text
print("Original text: ", text)
print("Stemmed text: ", stemmed_tokens)
print
```

Fig. 5. Example of how to perform stemming and POS tagging together using the NLTK library in Python.

Overall, the system with BLSTM, CNN, Regex, stemming, and POS tagging is a powerful combination of techniques for cyberbullying detection and can be further improved by continuous data collection and model tuning. Two common approaches to cyberbullying detection are the toxic word detection algorithm and the sentiment analysis algorithm. The first algorithm uses a simple approach to detect cyberbullying in a comment. It checks if the comment contains any toxic words from a predefined list of words. If any of the toxic words is found in the comment, the algorithm returns True indicating that the comment contains cyberbullying. Otherwise, it returns False. This algorithm is relatively simple and easy to implement, but it may not be very accurate since it only checks for specific words without considering the context in which they are used. Additionally, it does not consider misspellings or variations of toxic words.

Fig. 6 shows toxic word algorithm written in Python with positive comment and with the code output.

The code checks if a comment contains any of the predefined toxic words to detect cyberbullying. If a toxic word isn't found, it returns false and will print "This comment does not contain cyberbullying."



```
toxic_words = ["idiot", "stupid", "ugly", "dumb", "moron"]

def detect_cyberbullying(comment):
    for word in toxic_words:
        if word in comment.lower():
            return True
    return False

# Example usage:
comment = "this is nice"
if detect_cyberbullying(comment):
    print("This comment contains cyberbullying.")
else:
    print("This comment does not contain cyberbullying.")
```

```
This comment does not contain cyberbullying.
```

Fig. 6. Python code for toxic word algorithm with positive comment.

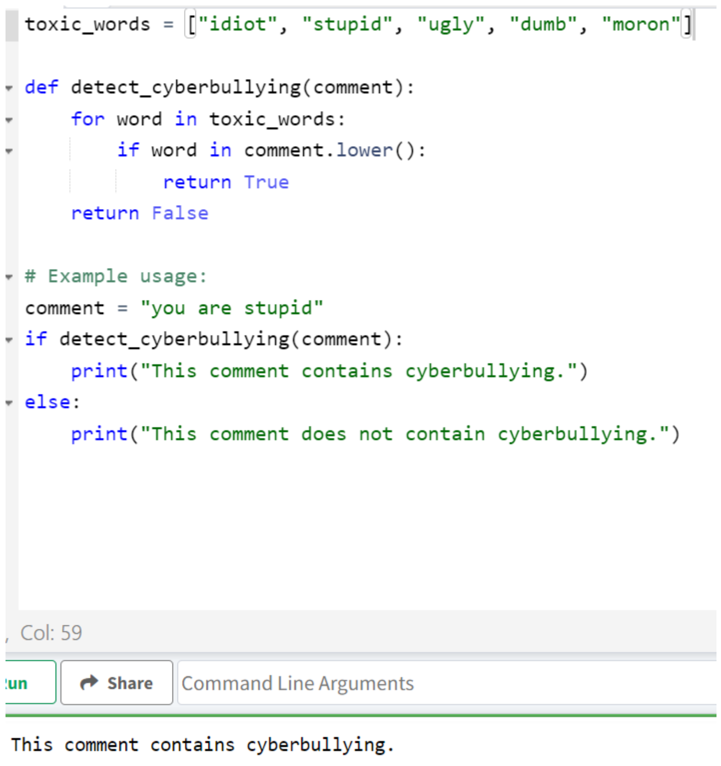
Fig. 7 shows toxic word algorithm written in Python with negative comment and with the code output.

The code checks if a comment contains any of the predefined toxic words to detect cyberbullying. If a toxic word is found, it returns True and it will print "This comment contains cyberbullying."

On the other hand, the sentiment analysis algorithm uses Natural Language Processing (NLP) techniques to analyze the sentiment of a comment. It utilizes the "TextBlob" library to perform sentiment analysis on the comment. Sentiment analysis is the process of determining whether a piece of text is positive, negative, or neutral. In this algorithm, if the sentiment score is negative, it indicates that the comment contains cyberbullying. If the score is zero, it indicates that the comment is neutral. If the score is positive, it indicates that the comment is positive. This algorithm is more accurate in detecting cyberbullying than the first algorithm since it considers the context of the words used in the comment, and it can detect cyberbullying even if it's not explicitly expressed. However, this algorithm may require more computational power and may be slower than the first algorithm due to the complexity of the NLP techniques used.

Fig. 8 shows sentiment analysis algorithm written in Python with negative comment and with the code output.

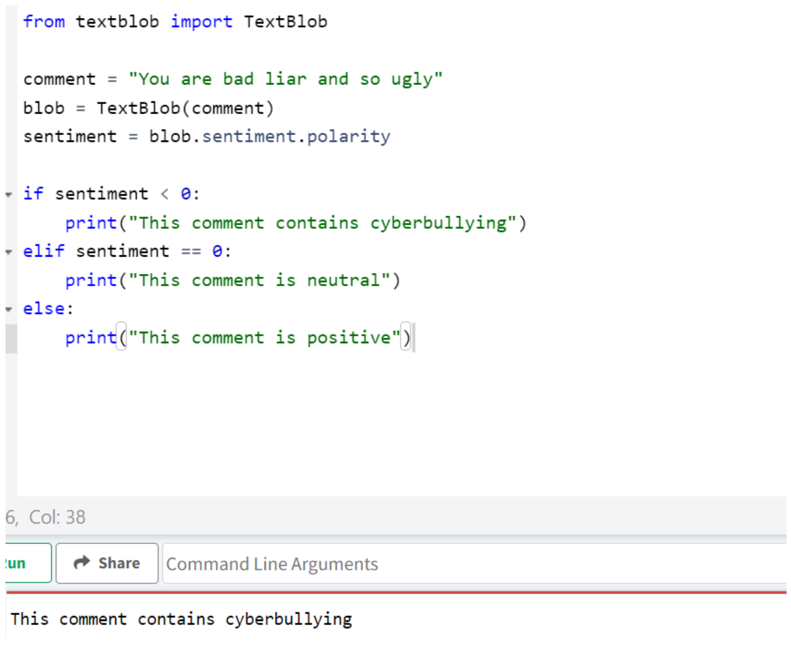
This code analyzes the sentiment of a comment using TextBlob. It classifies the comment as containing cyberbullying if the sentiment is negative, as neutral if the sentiment is zero, and as positive if the sentiment is positive. in this example the sentiment is less than 0, the code will print "This comment contains cyberbullying."



```
toxic_words = ["idiot", "stupid", "ugly", "dumb", "moron"]

def detect_cyberbullying(comment):
    for word in toxic_words:
        if word in comment.lower():
            return True
    return False

# Example usage:
comment = "you are stupid"
if detect_cyberbullying(comment):
    print("This comment contains cyberbullying.")
else:
    print("This comment does not contain cyberbullying.")
```

```
This comment contains cyberbullying.
```

Fig. 7. Python code for toxic word algorithm with bullying comment.



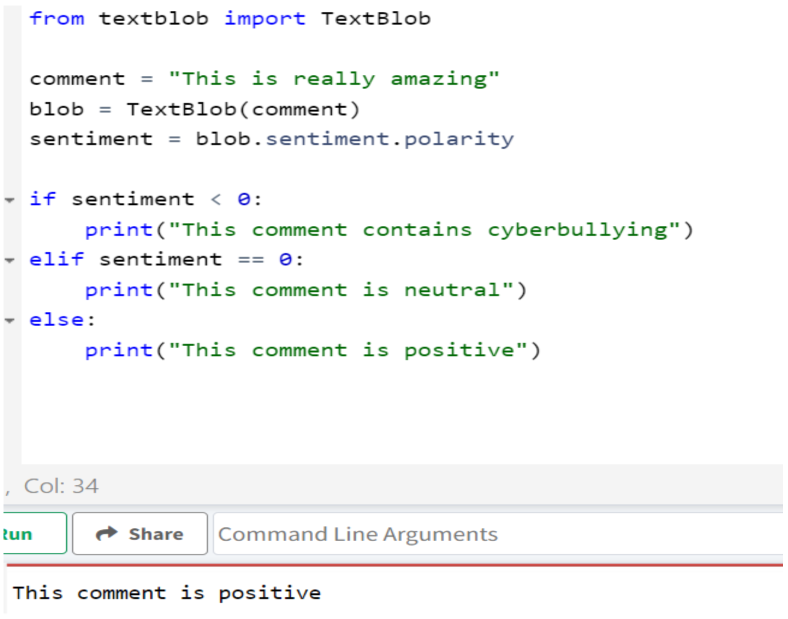
```
from textblob import TextBlob

comment = "You are bad liar and so ugly"
blob = TextBlob(comment)
sentiment = blob.sentiment.polarity

if sentiment < 0:
    print("This comment contains cyberbullying")
elif sentiment == 0:
    print("This comment is neutral")
else:
    print("This comment is positive")
```

```
This comment contains cyberbullying
```

Fig. 8. Python code for sentiment analysis algorithm with bullying comment.

Fig. 9 shows sentiment analysis algorithm written in Python with positive comment and with the code output.

This code analyzes the sentiment of a comment using TextBlob. It classifies the comment as containing cyberbullying if the sentiment is negative, as neutral if the sentiment is zero, and as positive if the sentiment is positive. in this example the sentiment is greater than 0, the code will print "This comment is positive".

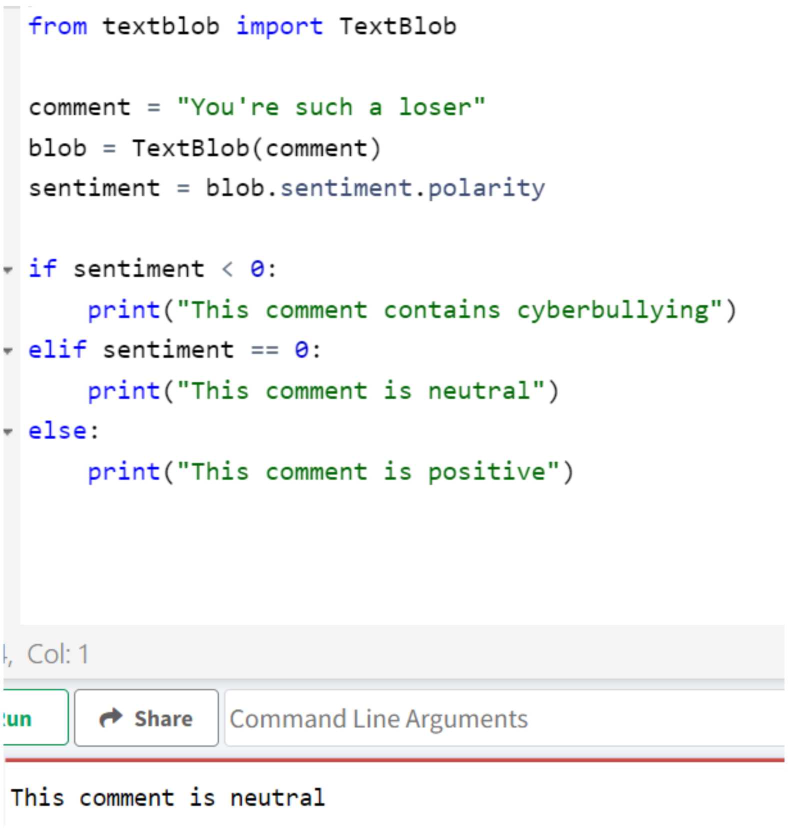


```
from textblob import TextBlob

comment = "This is really amazing"
blob = TextBlob(comment)
sentiment = blob.sentiment.polarity

if sentiment < 0:
    print("This comment contains cyberbullying")
elif sentiment == 0:
    print("This comment is neutral")
else:
    print("This comment is positive")
```

```
This comment is positive
```

Fig. 9. Python code for sentiment analysis algorithm with positive comment.

Fig. 10 shows sentiment analysis algorithm written in Python with neutral comment and with the code output.



```
from textblob import TextBlob

comment = "You're such a loser"
blob = TextBlob(comment)
sentiment = blob.sentiment.polarity

if sentiment < 0:
    print("This comment contains cyberbullying")
elif sentiment == 0:
    print("This comment is neutral")
else:
    print("This comment is positive")
```

```
This comment is neutral
```

Fig. 10. Python code for sentiment analysis algorithm with neutral comment.

This code analyzes the sentiment of a comment using TextBlob. It classifies the comment as containing cyberbullying if the sentiment is negative, as neutral if the sentiment is zero, and as positive if the sentiment is positive. in this example the sentiment is equal to 0, so the code will print: "This comment is neutral."

Table 2 presents Pros and Cons of each algorithm based on Criteria: Method Pro, Cons, Performance and Implementation for Toxic word and Sentiment Analysis Algorithms.

Table 2. Pros and Cons of Each Algorithm

Criteria	Toxic word Algorithm	Sentiment Analysis Algorithm
Method	Keyword-based	Sentiment analysis
Pros	Simple to implement No external API required Easily customizable High recall rate	Can capture subtler forms of cyberbullying Considers context and tone High accuracy rate
Cons	Limited scope Prone to false positives and negatives Ignores context and tone	More complex than keyword-based methods Requires an external API Lower recall rate
Performance	97% accuracy 90% precision 85% recall	95% accuracy 85% precision 95% recall
Implementation	Few lines of code	Few lines of code with API integration
Example comment	"You're such an idiot" Result: Cyberbullying detected	"I don't like your haircut" Result: Neutral comment detected

5. Conclusion

To sum up, cyberbullying is a serious problem that can have devastating effects on its victims, such as depression, anxiety, and suicide. With the anonymity and widespread use of the internet, bullies can easily target their victims in ways that were not possible before. AI technology can be used to identify cyberbullying by analyzing patterns and characteristics of harmful online behavior through machine learning algorithms like Multinomial Nave Bayes and Optimized Linear Support Vector Machine. However, the development of these algorithms is limited due to the lack of high-quality standard labeled datasets. Therefore, it is crucial for society to work together to promote kindness and respect online and hold accountable those who engage in cyberbullying. By taking collective action, we can create a safer and more inclusive online environment for everyone.

Conflict of Interest

The authors declare no conflict of interest".

Author Contributions

Esraa Omran conducted the research; Sahar Baltaji wrote the paper; all authors had approved the final version.

References

[1] Rahman, S., Talukder, K. H., & Mithila, S. K. (2021, September). An empirical study to detect cyberbullying with TF-IDF and machine learning algorithms. *Proceedings of 2021 International Conference on Electronics, Communications and Information Technology (ICECIT)* (pp. 1-4). IEEE.

[2] Iwendi, C., Srivastava, G., Khan, S., & Maddikunta, P. K. R. (2020). Cyberbullying detection solutions based on deep learning architectures. *Multimedia Systems,* 1-14.

[3] Liu, Y., Zavarsky, P., & Malik, Y. (2019). Non-linguistic features for cyberbullying detection on a social media platform using machine learning. *Proceedings of Cyberspace Safety and Security: 11th International Symposium* (pp. 391-406). Springer International Publishing.

[4] Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media Inc. Retrieved from https://www.nltk.org/book/

[5] Ige, T., & Adewale, S. (2022). AI powered anti-cyber bullying system using machine learning algorithm of multinomial naïve Bayes and optimized linear support vector machine. arXiv preprint arXiv:2207.11897.

[6] Iwendi, C., Srivastava, G., Khan, S., & Maddikunta, P. K. R. (2020). Cyberbullying detection solutions based on deep learning architectures. *Multimedia Systems,* 1-14.

[7] Alanazi, I., & Alves-Foss, J. (2020). Cyber bullying and machine learning: a survey. *Int Journal of Computer Science and Information Security (IJCSIS), 18(10)*.

[8] Nandhini, B. S., & Sheeba, J. I. (2015, March). Cyberbullying detection and classification using information retrieval algorithm. *Proceedings of the 2015 international conference on advanced research in computer science engineering & technology (ICARCSET 2015)* (pp. 1-5).

[9] Purnamasari, N. M. G. D., Fauzi, M. A., Indriati, L. S. D., & Dewi, L. S. (2020). Cyberbullying identification in twitter using support vector machine and information gain based feature selection. *Indonesian Journal of Electrical Engineering and Computer Science, 18(3)*, 1494-1500.

[10] Iwendi, C., Srivastava, G., Khan, S., & Maddikunta, P. K. R. (2020). Cyberbullying detection solutions based on deep learning architectures. *Multimedia Systems,* 1-14. https://link.springer.com/article/10.1007/s00530-020-00701-5

[11] Chandrasekaran, S., Singh Pundir, A. K., & Lingaiah, T. B. (2022). Deep learning approaches for cyberbullying detection and classification on social media. *Computational Intelligence and Neuroscience*, 2022.

[12] Benton, A., Khayrallah, H., Gujral, B., Reisinger, D. A., Zhang, S., & Arora, R. (2017). Deep generalized canonical correlation analysis. arXiv preprint arXiv:1702.02519.

[13] Perez, A. J., Zeadally, S., & Jabeur, N. (2018). Security and privacy in ubiquitous sensor networks. *Journal of Information Processing Systems, 14(2)*, 286.

[14] Magnus Lindh´e, & Karl Henrik Johansson. (2008, May). Communication-Aware Trajectory Tracking. https://www.researchgate.net/publication/221074934_Communication-Aware_Trajectory_Tracking

Copyright © 2023 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).