International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 2
Mar-Apr 2026, PP 31-43

Deepfake Detection On Social Media: Leveraging Deep Learning and Fasttext Embeddings for Identifying Machine-Generated Tweets

Dr. M. V. Subba Reddy[1], Nagularpu Hepsiba[2], Vaddemani Yagna Sree[3], Patnam Chandra Moulieswar[4], Shaik Asif[5]

[1]Professor, Department of CSE,Sai Rajeswari Institute of Technology, Proddatur

[2,3,4]UG Students, Department of CSE,Sai Rajeswari Institute of Technology, Proddatur

Abstract. Opinions on social media can be swayed thanks to new developments in natural language creation. The capacity of deep neural networks to generate content has also been enhanced via language modeling. Because of this, text-generative algorithms have improved to the point that attackers may train social bots to publish deepfakes that seem legitimate and sway public opinion. Reliable and precise deepfake social media message detection systems are required to address this issue. Keeping this in mind, the present research finds Twitter posts that are made by machines. Using a basic deep learning model and word embeddings, this study leverages the publicly available Tweepfake dataset to distinguish between human and bot-generated tweets. A standard convolutional neural network (CNN) architecture is trained to detect deepfake tweets using FastText word embeddings. In order to prove that the suggested strategy is better, this research compared it to various machine learning models that served as baselines. Here are some baseline approaches: FastText, Term Frequency, FastText subword embeddings, and Term Frequency Inverse Document Frequency. We also compare the proposed method to other deep learning models, such as CNN-LSTM and Long short-term memory (LSTM), to demonstrate its efficacy and utility in solving the problem. The CNN architecture, when combined with FastText embeddings, efficiently and correctly classifies twitter data with a 93% accuracy rate, according to the experimental results.

Keywords: Deepfake Detection, Social Media Analysis, Twitter Bot Detection, Deep Learning, Convolutional Neural Network (CNN), FastText Word Embeddings.

I. Introduction

The four main forms of user-generated content on social media platforms are text, images, audio, and video [1]. A bot uses techniques like deepfake, search-and-replace, and gap-filling text to run a phony social media account, like, share, and publish posts [2]. By analyzing incoming data, deep learning is able to learn feature representation. Deepfake, which consists of AI-generated text, images, audio, and video, has the potential to deceive [3]. By giving the idea that they were created by individuals, deepfake mul timedia have produced problems in politics [4]. In democracies in particular, the rapid

dissemination of misinformation through social media platforms facilitates the manipulation of public opinion and perception [5].The use of cyborg and sockpuppet accounts is employed for this purpose [6]. Social bots, which are entirely automated accounts, mimic human activity on social media platforms [7].The adversary can propagate false information more convincingly thanks to bots and new advancements in generative models based on natural language, such as Grover [9] and the GPT [8].

For example, in the 2017 Net Neutrality case, millions of duplicate comments impacted the Commission's decision to remove the rule [10]. It is important to consider the potential impact of complex transformer-based models on straightforward text manipulation approaches that could lead to erroneous conclusions. Some recent instances include automating the writing of blog posts and tweets to assess generating abilities using GPT-2 [11] and GPT-3 [12].

In response to queries posted on /r/AskReddit, a GPT-3 bot used the handle "/u/thegentlemetre" [13]. Regardless, the bot's remarks were generally innocuous. There has been no damage done, however this issue should make OpenAI careful about misusing GPT-3. In order to protect genuine information and democracy on social media, a sovereign system is required to recognize deepfake texts, which are machine-generated texts.

Objective

The goal of this study is to develop and evaluate a deep learning model for detecting social media deepfakes by identifying machine-generated tweets using FastText embeddings. In order to fight misinformation and increase consumer trust in online interactions, the study examines the linguistic features and contextual embeddings of tweets to differentiate between human-authored and machine-generated content.

II. Problem Statement

The integrity of information shared on social media platforms is at risk because to the fast development of natural language processing algorithms, which have enabled increasingly lifelike machine-generated text. With the use of sophisticated text-generative models, adversaries can construct deepfake posts that sway public opinion and conversation. Twitter is particularly susceptible to manipulation because of the rapid dissemination of misinformation and its impact on society.

In spite of growing concerns, there are currently no effective methods for identifying tweets that are generated by machines among the vast amount of user-generated content. It will need more precision and efficiency than what's now available to fight social bots' sophisticated tactics.

III. System Analysis

SYSTEM ARCHITECTURE:

International Journal of Statistics and Mathematical Research Trends
E-ISSN: 3108-0871, Volume 2 Issue 2
Mar-Apr 2026, PP 31-43

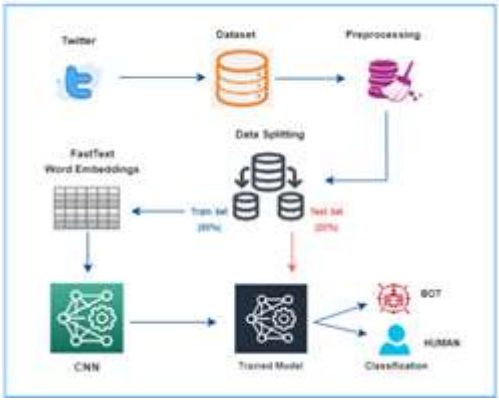


Fig.5.1.1 System architecture

Data Flow Diagram:
The DFD is sometimes known as a bubble chart. This straightforward visual formalism for depicting a system consists of three primary parts: data input, data processing, and data output.

Data flow diagrams (DFDs) are crucial to the modelling process. To model the parts of the system is the point of employing it. Everything that the system performs, any data it consumes, any third parties it communicates with, and any information that flows through it is included by this. Lastly, DFD reveals the data's evolution as it traverses the system. The changes that take place when data goes from an input to an output may be visually represented in a data flow diagram.

Bubble charts are sometimes known as data flow diagrams (DFDs). Any degree of abstraction may be shown in dynamic functional diagrams (DFDs). The complexity of its functions and the amount of data passing through them are two ways to classify DFD. The abbreviation UML, which stands for the Unified Modelling Language, is very useful. The Universal Modelling Language, or "UML" for short, is one rule for software engineering that centres on objects. The Object Management Group is in charge of supervising and developing the standard. The Unified Design Language's (UDL) primary objective is to provide a standard language for developing object-oriented software systems. Using its syntax and meta-model, modern UML is quite dependent.

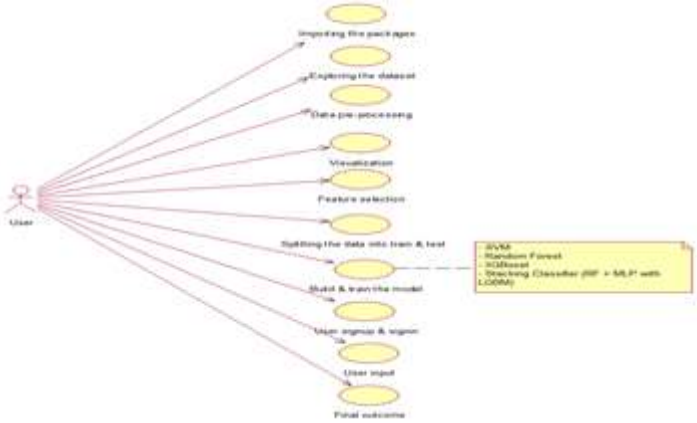
It is possible to include more procedures or methods into UML in the future. Any kind of artefact, from software systems to non-software systems to business models, may be more easily specified, visualised, produced, and recorded with the help of the Unified Modelling Language. Large and complicated systems may be represented using the Unified Modelling Language (UML), which offers a foundation for engineering best practices. Unified Modelling Language (UML) is a crucial component of creating object-oriented software or any application. When it comes to designing software projects, the Unified Modelling Language (UML) is primarily dependent on visual notations.

The original intent of the Unified Modelling Language (UML) was to accomplish the following:

1. Enable the development and sharing of practical models by providing a visual modelling language that is both expressive and easy to use.
2. Provide tools for customisation and expansion to enhance the adaptability of the main principles.

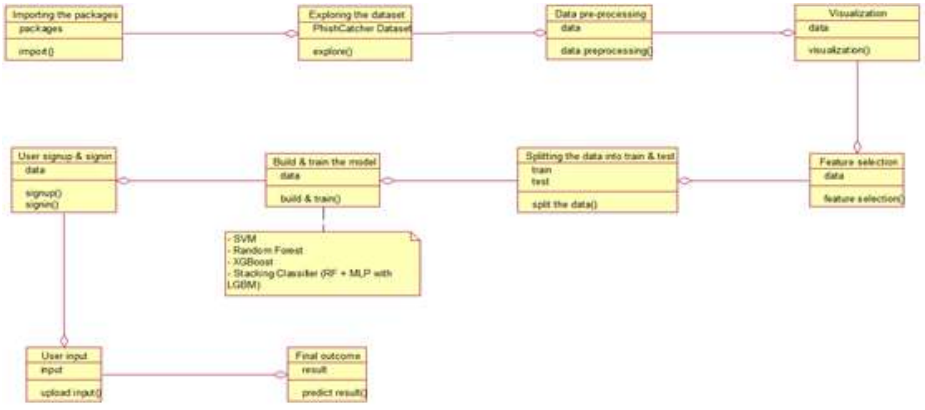
Thirdly, don't be closed off to other ways of thinking and doing things when it comes to growth, no matter what language you use.
Developing a strong formal foundation is the fourth stage in becoming a master modeller.

5. Require that a larger number of individuals acquire OO tools.



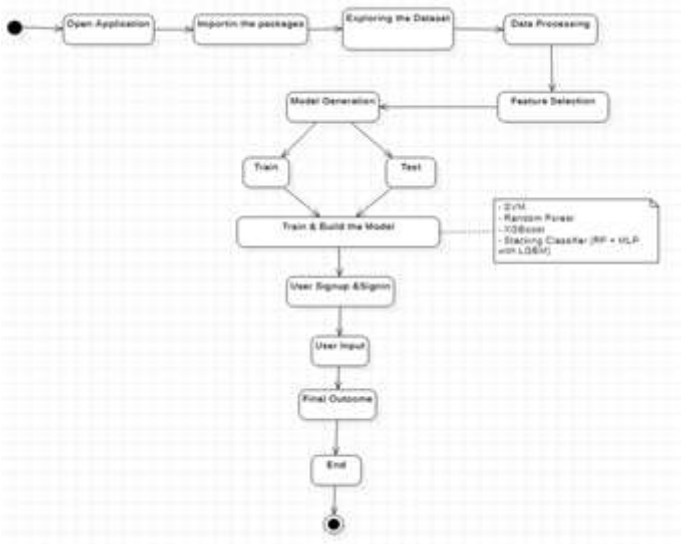
Class diagram:

It is possible to improve the use case diagram and provide a more detailed specification of the system's architecture by using the class diagram. The use case diagram specifies a group of actors and the class diagram details the connections between them. Your classes may use either a "is-a" or a "has-a" connection. It is possible that the classes shown in the class diagram may perform certain tasks. Classes accomplish their goals in part by means of the "methods" they provide. This aside, it's possible that distinct classes use different sets of "attributes" to distinguish themselves apart from one another.



Activity diagram:

International Journal of Statistics and Mathematical Research Trends
E-ISSN: 3108-0871, Volume 2 Issue 2
Mar-Apr 2026, PP 31-43

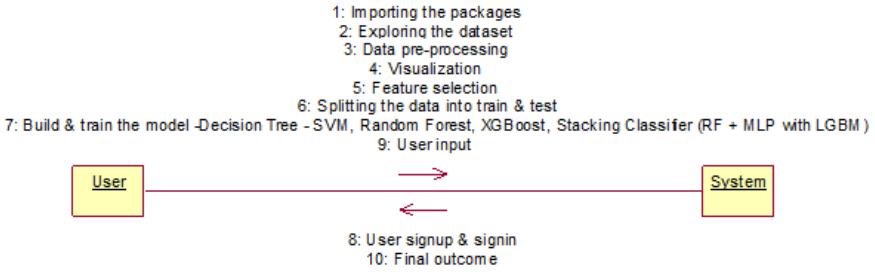


Sequence diagram:

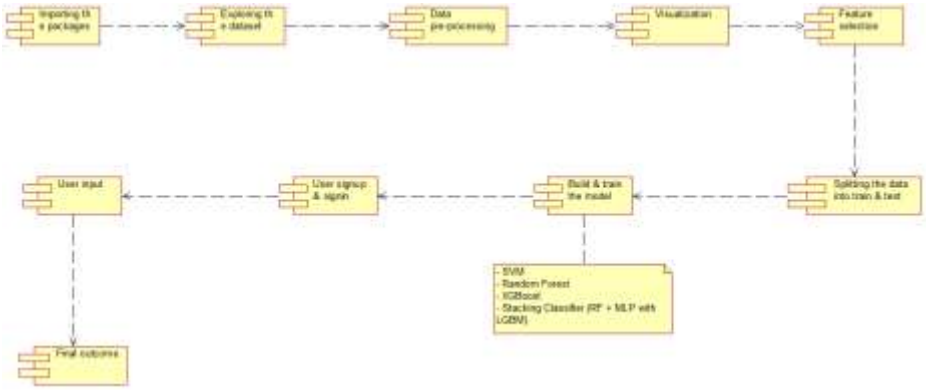
Draw a sequence diagram to show the interdependencies between the system's components. One distinguishing feature of a sequence diagram is the temporal ordering of occurrences. Because of this, interactions between objects may be shown in great detail and in a sequential fashion. The different components in the sequence diagram are able to interact with one another via the "messages" they use.

Collaboration diagram:

The relationships between many things may be shown in a cooperation diagram. The order of the interactions may be seen by looking at the numbered list of interactions. Every conceivable interaction between components may be shown in the cooperation diagram.



Component diagram:
The system's essential components are shown in the component diagram. This schematic provides an overview of the system by showing its primary parts and their interconnections. To better understand how a built or created system works, a component diagram may be very helpful.



IV. Implementation

MODULES:

- **Data loading:** using this module we are going to import the dataset.

Dataset Link:

https://www.kaggle.com/datasets/shauryapanpalia/cyberbullying-classification

Dataset Description:
This module allows us to create a 'train set' and a 'test set' from the information we have. The process is called model making. Naïve Bayes and logistic regression are used. The Idea of Bayes Probability and Fuzzy GA An amalgamation of LR fuzzy GA, RF, AB, and stacking classifiers into a single voting classifier. Accuracy of algorithms' predictions.

This module will be used to gather the login and registration information of users.
The users supplied data: To gather information for forecasts, you may use this module.

The projected final outcome is shown.

Using an ensemble approach to combine the predictions of many models led to a more accurate and reliable final forecast.

Improvements in performance, maybe as high as 99% accuracy, are possible with the addition of two additional ensemble methods: Voting Classifier and Stacking Classifier. Formulas for calculations [F]

One method of classification that relies on Bayes' Theorem is Naive Bayes, which assumes that predictors are independent. A Naive Bayes classifier, in its most basic form, takes for granted that there is no correlation between two features that share a class.

To solve binary classification issues, logistic regression determines the likelihood of a result, occurrence, or observation. It falls within the category of supervised machine learning. The model can only provide a yes/no, 0/1, or true/false result.

The Naive Bayes algorithm is among the most well-known and user-friendly approaches to machine learning classification, alongside Naive Bayes Fuzzy GA. One may calculate probabilities and conditional probabilities using Bayes' Theorem.

Because they lack both volume and mass, particles serve as the basic unit of measurement in the PSO algorithm. One such example is Naive Bayes Fuzzy PSO. The trajectories of particles are fine-tuned by constantly changing their velocities in response to their own and other particles' flight experiences within the search space.

An ML model called the Voting Classifier (AB + RF) accepts a number of models as input and returns the class that has the best chance of being selected.

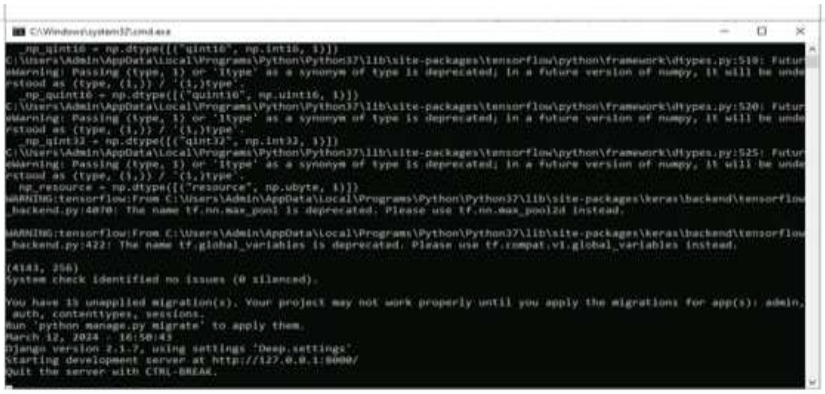
One ensemble approach to machine learning is stacking classifiers, which looks for methods to integrate the predictions of many successful ML models. Many Python programmers prefer to use the scikit-learn package when they need to build stacking ensembles.

SAMPLE CODE:

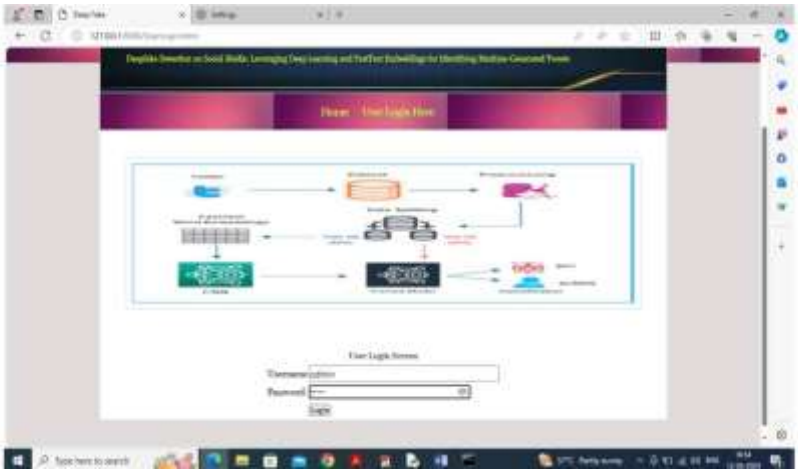
```
# Necessary imports
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
import time
import multiprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from collections import Counter
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
```

```
from imblearn.over_sampling import SMOTE, ADASYN
from sklearn.metrics import confusion_matrix, r2_score, mean_squared_error
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, accuracy_score, classification_report, precision_recall_curve
import warnings
warnings.filterwarnings("ignore")
df = pd.read_csv("/kaggle/input/network-anamoly-detection/Train.txt",sep=",",names=["duration","protocoltype","service","flag","srcbytes","dstbytes","land", "wrongfragment","urgent","hot","numfailedlogins","loggedin", "numcompromised","rootshell","suattempted","numroot","numfilecreations", "numshells","numaccessfiles","numoutboundcmds","ishostlogin",
"isguestlogin","count","srvcount","serrorrate", "srvserrorrate",
"rerrorrate","srvrerrorrate","samesrvrate", "diffsrvrate", "srvdiffhostrate","dsthostcount","dsthostsrvcount","dsthostsamesrvrate", "dsthostdiffsrvrate","dsthostsamesrcportrate",
"dsthostsrvdiffhostrate","dsthostserrorrate","dsthostsrvserrorrate",
"dsthostrerrorrate","dsthostsrvrerrorrate","attack", "lastflag"])
df.head()
df.drop(['land','urgent','numfailedlogins','numoutboundcmds'],axis=1,inplace=True)
df.isna().sum()
df.select_dtypes(exclude=[np.number])
df['attack'].loc[df['attack']!='normal']='attack'
le=LabelEncoder()
df['protocoltype']=le.fit_transform(df['protocoltype'])
df['service']=le.fit_transform(df['service'])
df['flag']=le.fit_transform(df['flag'])
df['attack']=le.fit_transform(df['attack'])
plt.figure(figsize=(20,15))
sns.heatmap(df.corr())
X=df.drop(['attack'],axis=1)
y=df['attack']
sns.countplot(df['attack'])
print("Class distribution: {}".format(Counter(y)))
scaler = StandardScaler()
scaler.fit(X)
X_transformed = scaler.transform(X)
lr=LogisticRegression()
lr.fit(X_transformed,y)
lr_pred=lr.predict(X_transformed)
lr_df=pd.DataFrame()
lr_df['actual']=y
lr_df['pred']=lr_pred
lr_df.head()
print(accuracy_score(y, lr_pred))
print(classification_report(y, lr_pred))
rf=RandomForestClassifier()
rf.fit(X_transformed,y)
rf_pred=rf.predict(X_transformed)
```

```
rf_df=pd.DataFrame()
rf_df['actual']=y
rf_df['pred']=rf_pred
rf_df.head()
print(accuracy_score(y, lr_pred))
print(classification_report(y, lr_pred))
test_df = pd.read_csv("/kaggle/input/network-anamoly-detection/Test.txt",sep=",",names=["duration","protocoltype","service","flag","srcbytes","dstbytes","land", "wrongfragment","urgent","hot","numfailedlogins","loggedin", "numcompromised","rootshell","suattempted","numroot","numfilecreations", "numshells","numaccessfiles","numoutboundcmds","ishostlogin",
"isguestlogin","count","srvcount","serrorrate", "srvserrorrate",
"rerrorrate","srvrerrorrate","samesrvrate", "diffsrvrate", "srvdiffhostrate","dsthostcount","dsthostsrvcount","dsthostsamesrvrate", "dsthostdiffsrvrate","dsthostsamesrcportrate",
"dsthostsrvdiffhostrate","dsthostserrorrate","dsthostsrvserrorrate",
"dsthostrerrorrate","dsthostsrvrerrorrate","attack", "lastflag"])
test_df.head()
test_df.select_dtypes(exclude=[np.number])
test_df['attack'].loc[test_df['attack']!='normal']='attack'
test_df['protocoltype']=le.fit_transform(test_df['protocoltype'])
test_df['service']=le.fit_transform(test_df['service'])
test_df['flag']=le.fit_transform(test_df['flag'])
test_df['attack']=le.fit_transform(test_df['attack'])
test_df.drop(['land','urgent','numfailedlogins','numoutboundcmds'],axis=1,inplace=True)
X_test=test_df.drop(['attack'],axis=1)
y_test=test_df['attack']
sns.countplot(test_df['attack'])
X_test_transformed = scaler.transform(X_test)
test_pred=rf.predict(X_test_transformed)
rf_test_df=pd.DataFrame()
rf_test_df['actual']=y_test
rf_test_df['pred']=test_pred
rf_test_df.head()
target_names=["attack","normal"]
print(classification_report(y_test, test_pred,target_names=target_names))
tn, fp, fn, tp = confusion_matrix(y_test, test_pred).ravel()
print("True Negatives:",tn)
print("False Positives:",fp)
print("False Negatives:",fn)
print("True Positives:",tp)
```



In above screen python server started and now open browser and enter URL as http://127.0.0.1:8000/index.html and press enter key to get below page



Username	Tweet	Account Type

International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 2
Mar-Apr 2026, PP 31-43

V. Conclusion

Even if it's challenging, deepfake text detection is vital in this era of manipulation and disinformation. To get around this problem, this study developed and evaluated a deep-fake text detecting technique. Machine learning, deep learning, and feature engineering are used to assess a set of tweets that contain both bot and human content. Useful tools include the well-known feature extraction algorithms Tf and TF-IDF, as well as the word embedding algorithms FastText and FastText subwords. The suggested method identified deepfake text with a 0.93 accuracy rate using CNN and FastText.

Several state-of-the-art transfer learning models are also compared to the proposed technique. From what we can tell from this research, CNN model structures are simpler, more efficient computationally, and far better at dealing with things that do not exist in our language. The findings of this study pave the way for further investigation into deepfake identification as well as practical implications. In the age of social mediastill plays a major role in Protecting actual information and democratic processes requires robust deepfake text detection algorithms, according to public opinion. Both [76] and [77] address the difficulties and potential of quantum machine learning, while the latter

provides a quantum method for dealing with deep fake text. Improved detection systems to counteract social media misinformation will be the focus of future research utilizing quantum NLP and other state-of-the-art methodologies.

References

1. J. P. Verma and S. Agrawal, ‘‘Big data analytics: Challenges and applica tions for text, audio, video, and social media data,’’ Int. J. Soft Comput., Artif. Intell. Appl., vol. 5, no. 1, pp. 41–51, Feb. 2016.
2. H. Siddiqui, E. Healy, and A. Olmsted, ‘‘Bot or not,’’ in Proc. 12th Int. Conf. Internet Technol. Secured Trans. (ICITST), Dec. 2017, pp. 462–463.
3. M. Westerlund, ‘‘The emergence of deepfake technology: A review,’’ Technol. Innov. Manage. Rev., vol. 9, no. 11, pp. 39–52, Jan. 2019.
4. J. Ternovski, J. Kalla, and P. M. Aronow, ‘‘Deepfake warnings for political videos increase disbelief but do not improve discernment: Evidence from two experiments,’’ Ph.D. dissertation, Dept. Political Sci., Yale Univ., 2021.
5. S. Vosoughi, D. Roy, and S. Aral, ‘‘The spread of true and false news online,’’ Science, vol. 359, no. 6380, pp. 1146–1151, Mar. 2018.
6. S. Bradshaw, H. Bailey, and P. N. Howard, ‘‘Industrialized disinformation: 2020 global inventory of organized social media manipulation,’’ Comput. Propaganda Project Oxford Internet Inst., Univ. Oxford, Oxford, U.K., Tech. Rep., 2021.
7. C.Grimme,M.Preuss,L.Adam,andH.Trautmann,‘‘Socialbots:Human like by means of human control?’’ Big Data, vol. 5, no. 4, pp. 279–293, Dec. 2017.
8. X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, ‘‘GPT understands, too,’’ 2021, arXiv:2103.10385.
9. R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, ‘‘Defending against neural fake news,’’ in Proc. 33rd Int. Conf. Neural Inf. Process. Syst. (NIPS), Dec. 2019, pp. 9054–9065, Art. no. 812.
10. L. Beckman, ‘‘The inconsistent application of internet regulations and suggestions for the future,’’ Nova Law Rev., vol. 46, no. 2, p. 277, 2021, Art. no. 2.
11. J.-S. Lee and J. Hsiang, ‘‘Patent claim generation by fine-tuning OpenAI GPT-2,’’ World Pat. Inf., vol. 62, Sep. 2020, Art. no. 101983.
12. R. Dale, ‘‘GPT-3: What’s it good for?’’ Natural Lang. Eng., vol. 27, no. 1, pp. 113–118, 2021.
13. W. D. Heaven, ‘‘A GPT-3 bot posted comments on Reddit for a week and no one noticed,’’ MIT Technol. Rev., Cambridge, MA, USA, Tech. Rep., Nov. 2020, p. 2020, vol. 24. [Online]. Available: www.technologyreview.com