# Fake News Monitoring and Anti-rumor System Using Deep Learning and Blockchain

*by Tristan Huang, Bangcheng Li, and Xin Wang*

## 1. Background

Hundreds of posts spreading misinformation about COVID-19 are being left online, according to a report from the Center for Countering Digital Hate. Most recently, fake news and rumors were cited as a negative contributing factor to the COVID-19 pandemic and BLM movements. Irresponsible individuals and organizations published misleading information causing catastrophic consequences to society. In the age of social media, the ability to spread false information has increased exponentially. Though technology has fostered spread fake news and rumors, it can also help to terminate.

Many technologies, such as AI-based fake new monitors and Blockchain-based platforms, have been developed to overcome the challenge. In 2017, Steve Huckle and Martin White examined how blockchains can be used to authenticate digital media by using an application capable of analyzing images and videos to verify the originality of media resources. In 2018, Thang Dinh and My Thai summarized the current uses of blockchain and AI technology and discussed potential future uses and integration into various industries.

In this project, we will summarize and combine the existing algorithms in deep learning and blockchain to develop a new anti-fake news and rumors system. We introduce the basic idea of anti-fake news system based on blockchain technology in section 2. In section 3 we utilize LSTM model as predictor to detect fake news with python codes. The contribution of our project is presented in section 4. We come up with a new blockchain-based

deep learning anti-fake news system, which can improve the ability to avoid fake news.

2. **Blockchain-based anti-fake news system**

   As an emerging technology, Blockchain technology has become an effective method to revolutionize information production and dissemination. Based on the traceability, transparency, and decentralization properties of blockchain, the problem of fake news can be solved easily. The blockchain technology-enabled platforms, like Facebook and Instagram, to check the authenticity of the information, in order to avoid people fall into the trap of fake news.

   Akash Takyar has indicated how the user personas involved in the Blockchain-based fake news platform. News Agencies will give news to the editors for publishing and upload necessary documents, such as name, news agency license, address proof, domain name, work permit, years of working certificates and etc. After the agencies send the content, the editors/ modifiers/ publishers will check the authenticity of the news and determine if it can be published. Journalists will upload the images or videos to the platform. The images or videos will be uploaded by journalists to the platform and be visible to everyone on the blockchain network. However, once the images or videos are saved on the blockchain, no one can delete or alter the data, which would increase information security. Editors will write and modify the news on the platform. When they upload their documents, third-party APIs like Trulio will check their background verification. A smart contract, a self-enforcing agreement embedded in computer code managed by a blockchain, would be a catalyst to provide ratings to the modifiers or editors. Moreover, the crowd auditors can find the complete source of certain news with the help of QR code and then identify if the news is real news or fake news. In the last stage, new data, videos, and images with their barcodes, content, and hash address will be saved on the public blockchain.

In conclusion, blockchain enables news platform could have the following benefits:

- **Transparency in the news**

  The whole world would be able to discover the authenticity of the news based on specific criteria defined in the smart contracts.

- **Traceability of the news**

  Tracing the authenticity of news all the time is possible through the blockchain. No one will be fallen into the track of fake news.

- **Decentralized Approach**

  Decentralization method can avoid single points of failure so that no partial failure would bring the entire system down.

- **Immutable Approach**

  No one can change or delete the news content, videos, or images, once all have been saved on the blockchain.

## 3. Deep learning-based fake news detection

### 3.1.LSTM predictions with performance

To build a fake news detection model, ISOT fake news dataset from Kaggle is applied. This dataset contains two types of articles: 21,417 real-news and 23,481 fake-news. The real news was collected by scraping articles from Reuters.com (News website), while the fake news was obtained from unauthorized websites that were flagged by Politifact (a fact-checking nonprofit in the USA) and Wikipedia. Although this dataset covers a variety of articles on different topics, most of the news in this dataset focuses on political and world affairs topics. In the LSTM model, we use a pre-trained embedding index from GoogleNews-vectors-negative300 to convert pure text to computable numbers, based on word embeddings technique.

Import all the modules we need.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.layers import Dense, Input, GlobalMaxPooling1D, LSTM
from keras.layers import Conv1D, MaxPooling1D, Embedding, Dropout, Activation, Flatten
from keras.models import Model
from keras.initializers import Constant
from keras.models import Sequential
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import *
from gensim.models.keyedvectors import KeyedVectors
```

Load the data.

```
# Use English stemmer.
word_stemmer = PorterStemmer()
wordnet_lemmatizer = WordNetLemmatizer()
# Load the data
real_news_df = pd.read_csv('True.csv')
fake_news_df = pd.read_csv('Fake.csv')
```

Remove missing values.

```
real_news_df['text'] = real_news_df['title'] + " " + real_news_df['text'] + " " +
real_news_df['subject']
fake_news_df['text'] = fake_news_df['title'] + " " + fake_news_df['text'] + " " +
real_news_df['subject']
real_news_df = real_news_df[real_news_df['text'].str.len() >= 3]
fake_news_df = fake_news_df[fake_news_df['text'].str.len() >=3]
```

Add two lables: 1 (real news) and 0 (fake news).

```
real_news_df['real_fact'] = 1
fake_news_df['real_fact'] = 0
```

Perform data cleaning.

```
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won\'t", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
```

```
    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase

def get_cleaned_data(input_data, mode='df'):
    stop = stopwords.words('english')
    input_df = ''
    if mode != 'df':
        input_df = pd.DataFrame([input_data], columns=['text'])
    else:
        input_df = input_data

    #lowercase the text
    input_df['text'] = input_df['text'].str.lower()
    input_df['text'] = input_df['text'].apply(lambda elem: decontracted(elem))
    #remove special characters
    input_df['text'] = input_df['text'].apply(lambda elem: re.sub(r"(@[A-Za-z0-9]+)|([^0-9A-Za-z
\t])|(\w+:\/\/\S+)|^rt|http.+?", "", elem))
    # remove numbers
    input_df['text'] = input_df['text'].apply(lambda elem: re.sub(r"\d+", "", elem))
    #remove stopwords
    input_df['text'] = input_df['text'].apply(lambda x: ' '.join([word.strip() for word in x.split() if
word not in (stop)]))
    #stemming, changes the word to root form
#     input_df['text'] = input_df['text'].apply(lambda words: [word_stemmer.stem(word) for word
in words])
    #lemmatization, same as stemmer, but language corpus is used to fetch the root form, so
resulting words make sense
#     more description @ https://www.datacamp.com/community/tutorials/stemming-
lemmatization-python
    input_df['text'] = input_df['text'].apply(lambda words:
(wordnet_lemmatizer.lemmatize(words)))
    return input_df
fake_news_df = get_cleaned_data(fake_news_df)
real_news_df = get_cleaned_data(real_news_df)
news_data_df = pd.concat([real_news_df, fake_news_df], ignore_index = True)
```

## Divide training set and test set.

```
MAX_SEQUENCE_LENGTH = 500
MAX_NUM_WORDS = 10000
EMBEDDING_DIM = 300
VALIDATION_SPLIT = 0.3

x_train,x_test,y_train,y_test =
train_test_split(news_data_df.text,news_data_df.real_fact,random_state = 42,
test_size=VALIDATION_SPLIT, shuffle=True)
```

## Vectorize the text samples into a two-dimensional tensor.

```
tokenizer = Tokenizer(num_words=MAX_NUM_WORDS)

tokenizer.fit_on_texts(x_train)

# Transforms each text in texts to a sequence of integers.
# So it basically takes each word in the text and replaces it with its corresponding integer value
from the word_index dictionary.
# sequences = tokenizer.texts_to_sequences(news_data_df.text)
tokenized_train = tokenizer.texts_to_sequences(x_train)
X_train = pad_sequences(tokenized_train, maxlen=MAX_SEQUENCE_LENGTH)

word_index = tokenizer.word_index

tokenized_test = tokenizer.texts_to_sequences(x_test)
X_test = pad_sequences(tokenized_test, maxlen=MAX_SEQUENCE_LENGTH)

def get_embeddings(path):
   # model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-
negative300', binary=True, limit=500000)
   wv_from_bin = KeyedVectors.load_word2vec_format(path, binary=True, limit=500000)
   #extracting word vectors from google news vector
   embeddings_index = {}
   for word, vector in zip(wv_from_bin.vocab, wv_from_bin.vectors):
      coefs = np.asarray(vector, dtype='float32')
      embeddings_index[word] = coefs
   return embeddings_index

embeddings_index = {}
embeddings_index = get_embeddings('GoogleNews-vectors-negative300\GoogleNews-vectors-
negative300.bin')
```

## Prepare embedding matrix.

```
vocab_size = len(tokenizer.word_index) + 1

embedding_matrix = np.zeros((vocab_size, EMBEDDING_DIM))
for word, i in word_index.items():
   try:
      embedding_vector = embeddings_index[word]
      embedding_matrix[i] = embedding_vector
   except KeyError:
      embedding_matrix[i]=np.random.normal(0,np.sqrt(0.25),EMBEDDING_DIM)
```

## Build the LSTM model.

```
def lstm_net1():
   model = Sequential()

   #Non-trainable embeddidng layer
   model.add(Embedding(vocab_size, output_dim=EMBEDDING_DIM,
weights=[embedding_matrix], input_length=MAX_SEQUENCE_LENGTH, trainable=False))

   model.add(LSTM(units=128 , return_sequences = True))
   model.add(Dropout(0.2))
   model.add(LSTM(units=64))
   model.add(Dropout(0.1))
   model.add(Dense(units = 32 , activation = 'relu'))
   model.add(Dense(1, activation='sigmoid'))
   model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

   return model
```

## Calculate training and testing accuracies.

```
accr_train = model2.evaluate(X_train,y_train)
print('Accuracy Train: {}'.format(accr_train[1]*100))
accr_test = model.evaluate(X_test,y_test)
print('Accuracy Test: {}'.format(accr_test[1]*100))
```

## Accuracy results:

```
937/937 [==============================] - 485s 518ms/step - loss: 0.1081 - accuracy: 0.9525
Accuracy Train: 95.24730443954468
402/402 [==============================] - 33s 82ms/step - loss: 0.0096 - accuracy: 0.9971
Accuracy Test: 99.71208572387695
```

Train the LSTM model.

```
#training an LSTM network
model2 = lstm_net1()

batch_size = 256
epochs = 8
history = model2.fit(X_train, y_train, batch_size = batch_size , validation_data = (X_test,y_test) ,
epochs = epochs)
```

Visualize LSTM confusion matrix.

```
pred = model2.predict_classes(X_test)
cf_matrix = confusion_matrix(y_test,pred)
sns.heatmap(cf_matrix, annot=True, fmt='g', xticklabels = ['Fake','Real'] , yticklabels =
['Fake','Real'])
```


Figure 1: LSTM confusion matrix

The confusion matrix is frequently used as a performance measurement for machine learning classification problem where output can be two or more classes. Thus, through figure 1, we can calculate with math that the recall is 0.957 and the precision is 0.942, which has indicated the superior performance of the LSTM prediction model.

### 3.2.LSTM for COVIEWED project

To fight against misinformation on the recent coronavirus pandemic, project COVIEWED is aiming to release a browser widget for labeling potential real/ fake claims on a website to ensure the reliability of the information. They collected different types of claims from internet sources and then trained the machine learning classifier using the data.

We choose three articles to test our LSTM model. Two of them are from reddit.com and the other one is from cbc.ca. Here, we only use one article to present how we test our model. We define a function to calculate the predicted value as below.

```
def get_pred_output(text_to_check):
   sequences = tokenizer.texts_to_sequences([text_to_check])
   data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)
   predicted_val = model.predict_classes(data)
   return predicted_val

reddit_2 = """A doctor at ……avoiding contact with people."
"""
text_check = reddit_2
pred = get_pred_output(text_check)
print('The predicted value is {} '.format(pred))
```

After testing all the three articles, we get the results indicated in the table below:

| Article ID | Article Source | Real Label | Predicted Label |
|---|---|---|---|
| **reddit_1** | reddit.com | Fake News | Real News |
| **reddit_2** | reddit.com | Fake News | Fake News |
| **cbc_1** | cbc.ca | Fake News | Real News |

Table 1: COVIEWED LSTM testing results

The predicting results still need to be improved to reach a satisfactory level.

## 4. How blockchain works with deep learning

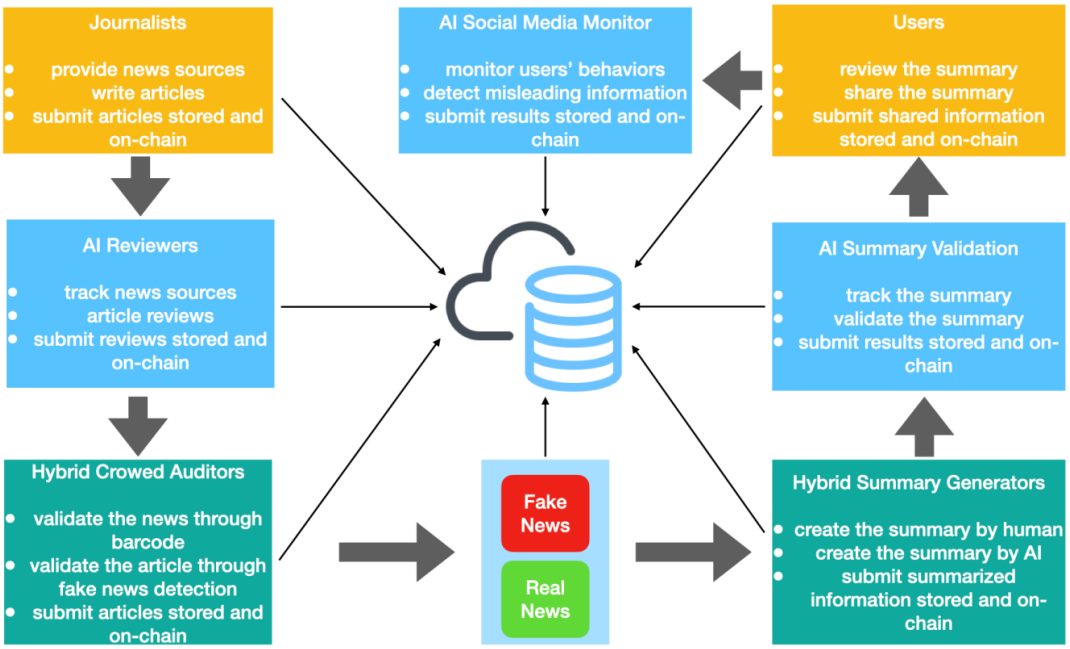


Figure 2: Deep learning on blockchain working flow

Compared to the blockchain-based anti-fake news system, the system can have more advantageous performance, when involving deep learning techniques into the blockchain. As shown in figure 2, in this system the starting point is from the journalists, who will provide the news sources, write articles, and submit articles stored and on-chain. Once the journalists finish their work, the news will be distributed to different AI reviewers. AI reviewers will go through the checking process for article reviewing and news source tracking. After that, hybrid crowed auditors will receive the data submitted by AI reviews on-chain. They will further perform validation through QR code and fake news detector. All the data will be classified as fake news and real news. Fake news will be filtered out, while real news can be sent to the next block. Two tasks are done by hybrid summary generators in this stage: creating the summary by humans and creating it by AI. The AI-summary can help us avoid deviations in the understanding of information due to human subjective knowledge. AI summary validation can give an alarm if the human-made summary extremely is opposite to the

original meaning of the news. Those news data passed by AI summary validation will be given to the users to read or spread on their social media on-chain. Eventually, the AI social media monitors will keep tracking users' behavior and see if they mislead any news they share. Therefore, this blockchain-based deep learning anti-fake news system can be carried out under the high adaptability and stability.

### 4.1.Deep learning as a server to review

We scraped three news samples from the thenextweb.com, which is a blockchain-based decentralized news website. Here, we only use one article to show how the server will review and detect the news.

```
dec_news_1 = """Donald Trump's ingenious idea …… Find it here on NYT."""
text_to_check = dec_news_1
pred = get_pred_output2(text_to_check)
print('Unseen real data prediction {} '.format(pred))
```

The testing results are shown in the table below.

| Article ID | Article Source | Real Label | Predicted Label |
|---|---|---|---|
| **dec_news_1** | thenextweb.com | Real News | Fake News |
| **dec_news_2** | thenextweb.com | Real News | Fake News |
| **dec_news_3** | thenextweb.com | Real News | Real News |

Table 2: Blockchain-based DL testing results on original news

### 4.2.Deep learning as a node to produce

As Figure 2 shown, AI summary validation can be conducted through deep learning technique on-chain, when running the following python scripts.

```
# importing libraries
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

# Input text - to summarize
text = dec_news_1
# Tokenizing the text
stopWords = set(stopwords.words("english"))
words = word_tokenize(text)
# Creating a frequency table to keep the
# score of each word
freqTable = dict()
for word in words:
    word = word.lower()
    if word in stopWords:
        continue
    if word in freqTable:
        freqTable[word] += 1
    else:
        freqTable[word] = 1



# Creating a dictionary to keep the score
# of each sentence
sentences = sent_tokenize(text)
sentenceValue = dict()
for sentence in sentences:
    for word, freq in freqTable.items():
        if word in sentence.lower():
            if sentence in sentenceValue:
                sentenceValue[sentence] += freq
            else:
                sentenceValue[sentence] = freq
sumValues = 0
for sentence in sentenceValue:
    sumValues += sentenceValue[sentence]

# Average value of a sentence from the original text
average = int(sumValues / len(sentenceValue))
# Storing sentences into our summary.
summary = ''
for sentence in sentences:
    if (sentence in sentenceValue) and (sentenceValue[sentence] > (1.2 * average)):
        summary += " " + sentence
print(summary)
dec_news_1 = summary
```

After validating three summarized news through LSTM model detection, we get the testing results as below:

| Article ID | Article Source | Real Label | Predicted Label |
|---|---|---|---|
| **sum_dec_news_1** | AI summary for blockchain | Real News | Fake News |
| **sum_dec_news_2** | AI summary for blockchain | Real News | Fake News |
| **sum_dec_news_3** | AI summary for blockchain | Real News | Real News |

Table 3: Blockchain-based DL testing results on summarized news

## 5. Discussion

While the Blockchain-based DL anti-fake news system can be used to control the spread of the misinformation, from the technical level there are still unresolved issues.

Even though this anti-fake news system has presented its reliable performance, we still can see some bias in the output result of the detection. We obtained Blockchain-based certified news through thenextweb.com, however, we cannot confirm whether the news information on the website is authorized by the blockchain. In the next stage of this project, we need to find out more dependable Blockchain-based news sources.

In addition, from Table 2 and 3, we can see that the predicting results will still be biased, caused by the unprofessional writing expression. In Table 2, the news article dec_news_3 has more professional and formal writing expression, thus, its predicted value is consistent with its real value.

Furthermore, the training set that we used to train the LSTM model is collected by Hadeer Ahmed, Issa Traore, and Sherif Saad in 2017, however, it's not a Blockchain-based database. In our further research, we will find out or even build a truly reliable news database verified by blockchain.

**Want to join us to develop more? Please visit [intelrabbit.com](intelrabbit.com).**

# 运用深度学习与区块链技术构建假新闻探测与反谣言系统

## 1背景

根据反对数字仇恨中心的一份报告，数百条散布有关COVID-19的错误信息的帖子被留在网上。 最近，虚假新闻和谣言被认为是导致COVID-19大流行和黑人运动的负面因素。 不负责任的个人和组织发布了误导性信息，给社会造成了灾难性的后果。 在社交媒体时代，传播虚假信息的能力呈指数增长。 尽管技术促进了虚假新闻和谣言的传播，但它也可以帮助将其终止。

为了克服挑战，许多技术已经被开发，例如基于AI的假新新闻监控器和基于区块链的平台。 在2017年，史蒂夫·休克（Steve Huckle）和马丁·怀特（Martin White）研究了如何通过使用能够分析图像和视频以验证媒体资源的原创性的应用程序，将区块链用于认证数字媒体。 2018年，Thang Dinh和My Thai总结了区块链和AI技术的当前用途，并讨论了未来的潜在用途以及与各个行业的集成。

在这个项目中，我们将总结并结合深度学习和区块链中的现有算法，以开发新的反假新闻和谣言系统。 在第2节中，我们介绍了基于区块链技术的反假新闻系统的基本思想。在第3节中，我们使用LSTM模型作为预测器，以python代码检测假新闻。 第4节介绍了我们项目的贡献。我们提出了一个新的基于区块链的深度学习反假新闻系统，该系统可以提高避免假新闻的能力。

## 2基于区块链的反假新闻系统

作为一种新兴技术，区块链技术已经成为变革信息生产和传播的有效方法。 基于区块链的可追溯性，透明性和分散性，可以轻松解决假新闻问题。 区块链技术使Facebook和Instagram等平台能够检查信息的真实性，从而避免人们陷入假新闻陷阱。

指出了用户如何参与到基于区块链的假新闻平台。新闻社将新闻发布给编辑，以发布和上传必要的文件，例如姓名，新闻社许可证，地址证明，域名，工作许可证，工作年限等。新闻社发送内容后，编辑/修改人/发布者将检查新闻的真实性，并确定是否可以发布新闻。记者将图像或视频上传到平台。图片或视频将由记者上传到平台，并且对区块链网络上的所有人可见。但是，一旦图像或视频保存在区块链上，就没有人可以删除或更改数据，这将提高信息安全性。编辑将在平台上撰写和修改新闻。当他们上传文档时，第三方API（例如Trulio）将检查其后台验证。智能合约是嵌入在由区块链管理的计算机代码中的一种自我执行的协议，它将成为向修改者或编辑者提供评级的催化剂。此外，人群审计员可以借助QR码找到特定新闻的完整来源，然后确定新闻是真实新闻还是虚假新闻。在最后阶段，新数据，视频和图像及其条形码，内容和哈希地址将保存在公共区块链上。

总之，区块链使新闻平台具有以下好处：

- **新闻透明**

  全世界都可以基于智能合约中定义的特定标准来发现新闻的真实性。

- **新闻的可追溯性**

  通过区块链可以始终追踪新闻的真实性。 没有人会落入虚假新闻的轨道。

- **去中心化方法**

  去中心化方法可以避免单点故障，因此没有部分故障会导致整个系统瘫痪。

- **不可变式**

  一旦所有内容都已保存在区块链上，任何人都无法更改或删除新闻内容，视频或图像。

## 3基于深度学习的虚假新闻检测

### 1. LSTM预测性能

为了构建假新闻检测模型，应用了Kaggle的ISOT假新闻数据集。 该数据集包含两种类型的文章：21,417个真实新闻和23,481个虚假新闻。 真实新闻是通过从Reuters.com（新闻网站）上抓取文章收集的，而虚假新闻是从未经授权的网站获得的，这些网站由Politifact（在美国进行事实核查的非营利组织）和Wikipedia进行了标记。 尽管此数据集涵盖了有关不同主题的各种文章，但该数据集中的大多数新闻都集中在政治和世界事务主题上。 在LSTM模型中，我们使用基于词嵌入技术的GoogleNews-vectors-negative300中的预训练嵌入索引将纯文本转换为可计算的数字。

导入我们需要的所有模块。

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.layers import Dense, Input, GlobalMaxPooling1D, LSTM
from keras.layers import Conv1D, MaxPooling1D, Embedding, Dropout, Activation, Flatten
from keras.models import Model
from keras.initializers import Constant
from keras.models import Sequential
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import *
from gensim.models.keyedvectors import KeyedVectors
```

导入数据。

```
# Use English stemmer.
word_stemmer = PorterStemmer()
wordnet_lemmatizer = WordNetLemmatizer()
# Load the data
real_news_df = pd.read_csv('True.csv')
fake_news_df = pd.read_csv('Fake.csv')
```

去除缺失值。

```
real_news_df['text'] = real_news_df['title'] + " " + real_news_df['text'] + " " +
real_news_df['subject']
fake_news_df['text'] = fake_news_df['title'] + " " + fake_news_df['text'] + " " +
real_news_df['subject']
real_news_df = real_news_df[real_news_df['text'].str.len() >= 3]
fake_news_df = fake_news_df[fake_news_df['text'].str.len() >=3]
```

加标签: 1 (真新闻) 和 0 (假新闻)。

```
real_news_df['real_fact'] = 1
fake_news_df['real_fact'] = 0
```

实施数据清洗。

```
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won\'t", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase

def get_cleaned_data(input_data, mode='df'):
    stop = stopwords.words('english')
    input_df = ''
    if mode != 'df':
        input_df = pd.DataFrame([input_data], columns=['text'])
    else:
        input_df = input_data

    #lowercase the text
    input_df['text'] = input_df['text'].str.lower()
    input_df['text'] = input_df['text'].apply(lambda elem: decontracted(elem))
    #remove special characters
    input_df['text'] = input_df['text'].apply(lambda elem: re.sub(r"(@[A-Za-z0-9]+)|([^0-9A-Za-z
\t])|(\w+:\/\/\S+)|^rt|http.+?", "", elem))
    # remove numbers
    input_df['text'] = input_df['text'].apply(lambda elem: re.sub(r"\d+", "", elem))
    #remove stopwords
    input_df['text'] = input_df['text'].apply(lambda x: ' '.join([word.strip() for word in x.split() if
word not in (stop)]))
    #stemming, changes the word to root form
#     input_df['text'] = input_df['text'].apply(lambda words: [word_stemmer.stem(word) for word
in words])
    #lemmatization, same as stemmer, but language corpus is used to fetch the root form, so
resulting words make sense
```

```
#     more description @ https://www.datacamp.com/community/tutorials/stemming-
lemmatization-python
   input_df['text'] = input_df['text'].apply(lambda words:
(wordnet_lemmatizer.lemmatize(words)))
   return input_df
fake_news_df = get_cleaned_data(fake_news_df)
real_news_df = get_cleaned_data(real_news_df)
news_data_df = pd.concat([real_news_df, fake_news_df], ignore_index = True)
```

分出训练集和测试集。

```
MAX_SEQUENCE_LENGTH = 500
MAX_NUM_WORDS = 10000
EMBEDDING_DIM = 300
VALIDATION_SPLIT = 0.3

x_train,x_test,y_train,y_test =
train_test_split(news_data_df.text,news_data_df.real_fact,random_state = 42,
test_size=VALIDATION_SPLIT, shuffle=True)
```

将文本样本矢量化为二维张量。

```
tokenizer = Tokenizer(num_words=MAX_NUM_WORDS)

tokenizer.fit_on_texts(x_train)

# Transforms each text in texts to a sequence of integers.
# So it basically takes each word in the text and replaces it with its corresponding integer value
from the word_index dictionary.
# sequences = tokenizer.texts_to_sequences(news_data_df.text)
tokenized_train = tokenizer.texts_to_sequences(x_train)
X_train = pad_sequences(tokenized_train, maxlen=MAX_SEQUENCE_LENGTH)

word_index = tokenizer.word_index

tokenized_test = tokenizer.texts_to_sequences(x_test)
X_test = pad_sequences(tokenized_test, maxlen=MAX_SEQUENCE_LENGTH)

def get_embeddings(path):
   # model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-
negative300', binary=True, limit=500000)
   wv_from_bin = KeyedVectors.load_word2vec_format(path, binary=True, limit=500000)
   #extracting word vectors from google news vector
   embeddings_index = {}
   for word, vector in zip(wv_from_bin.vocab, wv_from_bin.vectors):
      coefs = np.asarray(vector, dtype='float32')
      embeddings_index[word] = coefs
   return embeddings_index

embeddings_index = {}
```

```
embeddings_index = get_embeddings('GoogleNews-vectors-negative300\GoogleNews-vectors-negative300.bin')
```

准备嵌入矩阵。

```
vocab_size = len(tokenizer.word_index) + 1

embedding_matrix = np.zeros((vocab_size, EMBEDDING_DIM))
for word, i in word_index.items():
    try:
        embedding_vector = embeddings_index[word]
        embedding_matrix[i] = embedding_vector
    except KeyError:
        embedding_matrix[i]=np.random.normal(0,np.sqrt(0.25),EMBEDDING_DIM)
```

建立LSTM模型。

```
def lstm_net1():
    model = Sequential()

    #Non-trainable embeddidng layer
    model.add(Embedding(vocab_size, output_dim=EMBEDDING_DIM,
weights=[embedding_matrix], input_length=MAX_SEQUENCE_LENGTH, trainable=False))

    model.add(LSTM(units=128 , return_sequences = True))
    model.add(Dropout(0.2))
    model.add(LSTM(units=64))
    model.add(Dropout(0.1))
    model.add(Dense(units = 32 , activation = 'relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    return model
```

计算训练集和测试集的准确度。

```
accr_train = model2.evaluate(X_train,y_train)
print('Accuracy Train: {}'.format(accr_train[1]*100))
accr_test = model.evaluate(X_test,y_test)
print('Accuracy Test: {}'.format(accr_test[1]*100))
```

准确度结果：

```
937/937 [==============================] - 485s 518ms/step - loss: 0.1081 - accuracy: 0.9525
Accuracy Train: 95.24730443954468
402/402 [==============================] - 33s 82ms/step - loss: 0.0096 - accuracy: 0.9971
Accuracy Test: 99.71208572387695
```

训练LSTM 模型。

```
#training an LSTM network
model2 = lstm_net1()

batch_size = 256
epochs = 8
history = model2.fit(X_train, y_train, batch_size = batch_size , validation_data = (X_test,y_test) , epochs = epochs)
```

可视化LSTM混淆矩阵。

```
pred = model2.predict_classes(X_test)
cf_matrix = confusion_matrix(y_test,pred)
sns.heatmap(cf_matrix, annot=True, fmt='g', xticklabels = ['Fake','Real'] , yticklabels = ['Fake','Real'])
```


图 1: LSTM 混淆矩阵

混淆矩阵经常用作机器学习分类问题的性能度量，其中输出可以是两个或更多类。 因此，通过图1，我们可以通过数学计算得出召回率为0.957，精度为0.942，这表明LSTM预测模型具有优越的性能。

## 2. LSTM 用于 COVIEWED 项目

为了避免有关最近的冠状病毒大流行的错误信息，项目COVIEWED旨在发布一个浏览器窗口小部件，以在网站上标记潜在的真实/虚假声明，以确保信息的可靠性。 他们从互联网来源收集了不同类型的信息，然后使用这些数据训练了机器学习分类器。

我们选择三篇文章来测试我们的LSTM模型。 其中两个来自reddit.com，另一个来自cbc.ca。 在这里，我们仅使用一篇文章介绍如何测试模型。 我们定义一个函数来计算预测值，如下所示。

```
def get_pred_output(text_to_check):
   sequences = tokenizer.texts_to_sequences([text_to_check])
   data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)
   predicted_val = model.predict_classes(data)
   return predicted_val

reddit_2 = """A doctor at ……avoiding contact with people."
"""
text_check = reddit_2
pred = get_pred_output(text_check)
print('The predicted value is {} '.format(pred))
```

在测试了所有三篇文章之后，我们得到下表中指示的结果：

| Article ID | Article Source | Real Label | Predicted Label |
|---|---|---|---|
| **reddit_1** | reddit.com | Fake News | Real News |
| **reddit_2** | reddit.com | Fake News | Fake News |
| **cbc_1** | cbc.ca | Fake News | Real News |

图 1: COVIEWED LSTM 模型测试结果

预测结果有待提高以达到满意程度。

## 4. 区块链如何与深度学习一同工作

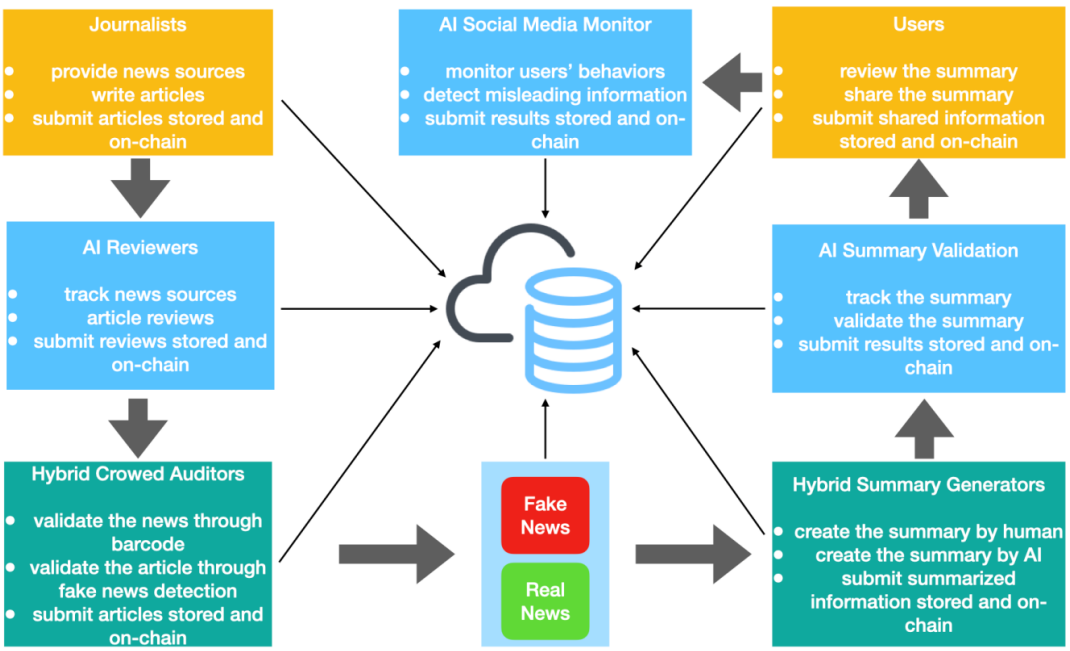


图 2: 基于区块链的深度学习工作执行图

与基于区块链的反假新闻系统相比，当将深度学习技术纳入区块链时，该系统可以具有更优越的性能。如图2所示，在该系统中，起点是新闻记者，新闻记者将提供新闻来源，撰写文章并提交存储在区块链上。记者完成工作后，新闻将分发给不同的AI审阅者。 AI审阅者将执行审阅和新闻来源跟踪的检查过程。在此之后，混合式审核员将接收AI评论链上提交的数据。他们将通过QR码和假新闻检测器进一步执行验证。所有数据将被分类为虚假新闻和真实新闻。假新闻将被过滤掉，而真实新闻可以发送到下一个区块。混合摘要生成器在此阶段完成了两项任务：由人创建摘要，由AI创建摘要。 AI摘要可以帮助我们避免由于人类主观知识而导致的信息理解偏差。如果人为摘要与新闻的原始含义极为相反，则AI摘要验证可以发出警报。通过AI摘要验证传递的那些新闻数据将提供给用户，以在链上的社交媒体上阅读或传播。最终，AI社交媒体监控器将继续跟踪用户的行为，并查看他们是否误导了

他们分享的任何新闻。因此，这种基于区块链的深度学习反假新闻系统可以在高度适应性和稳定性下进行。

### 4.1.深度学习作为服务器进行审阅

我们从thenextweb.com（这是一个基于区块链的分散式新闻网站）中抓取了三个新闻样本。 在这里，我们仅使用一篇文章来说明服务器将如何查看和检测新闻。

```
dec_news_1 = """Donald Trump’s ingenious idea …… Find it here on NYT."""
text_to_check = dec_news_1
pred = get_pred_output2(text_to_check)
print('Unseen real data prediction {} '.format(pred))
```

测试结果如下表所示。

| Article ID | Article Source | Real Label | Predicted Label |
|---|---|---|---|
| **dec_news_1** | thenextweb.com | Real News | Fake News |
| **dec_news_2** | thenextweb.com | Real News | Fake News |
| **dec_news_3** | thenextweb.com | Real News | Real News |

表 2: 基于区块链的深度学习对原文的测试结果

### 4.2.深度学习作为产生节点

如图2所示，当运行以下python脚本时，可以通过链上的深度学习技术来进行AI摘要验证。

```
# importing libraries
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

# Input text - to summarize
text = dec_news_1
# Tokenizing the text
stopWords = set(stopwords.words("english"))
words = word_tokenize(text)
# Creating a frequency table to keep the
# score of each word
freqTable = dict()
for word in words:
    word = word.lower()
    if word in stopWords:
        continue
    if word in freqTable:
        freqTable[word] += 1
    else:
        freqTable[word] = 1



# Creating a dictionary to keep the score
# of each sentence
sentences = sent_tokenize(text)
sentenceValue = dict()
for sentence in sentences:
    for word, freq in freqTable.items():
        if word in sentence.lower():
            if sentence in sentenceValue:
                sentenceValue[sentence] += freq
            else:
                sentenceValue[sentence] = freq
sumValues = 0
for sentence in sentenceValue:
    sumValues += sentenceValue[sentence]

# Average value of a sentence from the original text
average = int(sumValues / len(sentenceValue))
# Storing sentences into our summary.
summary = ''
for sentence in sentences:
    if (sentence in sentenceValue) and (sentenceValue[sentence] > (1.2 * average)):
        summary += " " + sentence
print(summary)
dec_news_1 = summary
```

在通过LSTM模型检测验证了三个摘要消息后，我们得到的测试结果如下：

| Article ID | Article Source | Real Label | Predicted Label |
|---|---|---|---|
| **sum_dec_news_1** | AI summary for blockchain | Real News | Fake News |
| **sum_dec_news_2** | AI summary for blockchain | Real News | Fake News |
| **sum_dec_news_3** | AI summary for blockchain | Real News | Real News |

表 3: 基于区块链的深度学习对已概括新闻的测试结果

## 5. 讨论

虽然可以使用基于区块链的深度学习反假新闻系统来控制错误信息的传播，但是从技术层面来看，仍然存在未解决的问题。

即使此防伪新闻系统已经表现出其可靠的性能，我们仍然可以看到检测输出结果中存在一些偏差。我们通过thenextweb.com获得了基于区块链的认证新闻，但是，我们无法确认网站上的新闻信息是否由区块链授权。在本项目的下一阶段，我们需要找到更可靠的基于区块链的新闻资源。

另外，从表2和表3中，我们可以看到由于非专业的写作表达而导致的预测结果仍然存在偏差。在表2中，新闻文章dec_news_3具有更专业，更正式的表达方式，因此，其预测值与实际值一致。

此外，我们用来训练LSTM模型的训练集由Hadeer Ahmed，Issa Traore和Sherif Saad于2017年收集，但是，它不是基于区块链的数据库。在我们

的进一步研究中，我们将发现甚至建立一个真正可通过区块链验证的可靠新闻数据库。

**想要加入我们的开发团队？请访问intelrabbit.com**