

## **1. INTRODUCTION**

Bursting various social networking sites (twitter, Facebook. .etc.) leads to a huge impact on news and information propagation. There are 3.20 billion social media geeks around the world. However, in spite of the net access or the age of the person 50% of the world population has a routine of using social networks. Using a social network, the public connected to the world has superior and faster access to current access and can help others. After all, it has many benefits, there are challenging tasks verifying the truthfulness of posts and recognizing the user who initiates and propagates a rumor or fake on the social media. Initial posts are frequently changed by unauthorized users and spread quickly around the internet. Due to these successive alterations, the meaning of the original post changes in a confused way. In the huge diffusion of such news, AI plays an important role where the automatic programs hat share fake news with a very higher frequency than usual social media Geeks. Fake news and rumors are used a frequently in the literature but there is a difference in the terms. Fake news is a intentionally miss leading article. Such stories are later Proven as rumors if those are false. Spreading such wrong information misled the population and approved severe incidences such as violence. Therefore, there is a need for a more appropriate and automatic rumors detection system.

Some effective methods are required to fight the spread of such news, which builds fear and anxiety among society. Many fact-checking websites such as Politick, Snopes, Fact Check work for debunking rumors or fake news. Also, there are crowdsourcing-based fact-checking sites Twitter and Facebook. The rapid circulation of stories can create chaos within the society if not handled early. In the case of dependable events, the consequences can be frightful. News verification through manual efforts is time taking. Recognition of rumors and rumors sources can control rumors dissemination. Though few researchers combined time-dependent characteristics with other features such as user, content-based, they use aggregate or fraction values for such features. These bulk values ignored many important components associated with an individual post. This research uses the meaningful post-wise features from various categories such as user based, content-based, lexical and post-based features using different deep learning models.

## **1.1 PROBLEM STATEMENT**

Design and develop rumor detection system to verify the truthfulness of post or tweets uploaded by users.

## **1.2 PROBLEM DESCRIPTION**

The rumors detection in social sites formally presented as the event driven sequence of posts given as input to the offered model identifies whether the event is rumors or non-rumors. The event is any incident that happened around us and informed through news, messages, likewise news related bomb blasts, political statements, focused organizations, etc. The input data contains a set of events  $E = \{e_1, e_2, \dots, e_N\}$ , where  $N$  is the total number of events. Each event includes  $n$  posts, as  $e_1 = \{p_1, p_2, \dots, p_n\}$ , where  $n$  is varying in size and  $p$  contains messages or tweets related to the event. Together with the posts, other aspects are extracted, such as,  $UserVect = \{u_1, u_2, \dots, u_n\}$ , where  $u_1 \dots u_n$  are user features like user .and  $LexVect = \{l_1, l_2, \dots, l_n\}$ , where  $l_1 \dots l_n$  are lexical features. The offered model's goal is to check whether the Post is a rumour or not by considering various post related features.

## **1.3 OBJECTIVE**

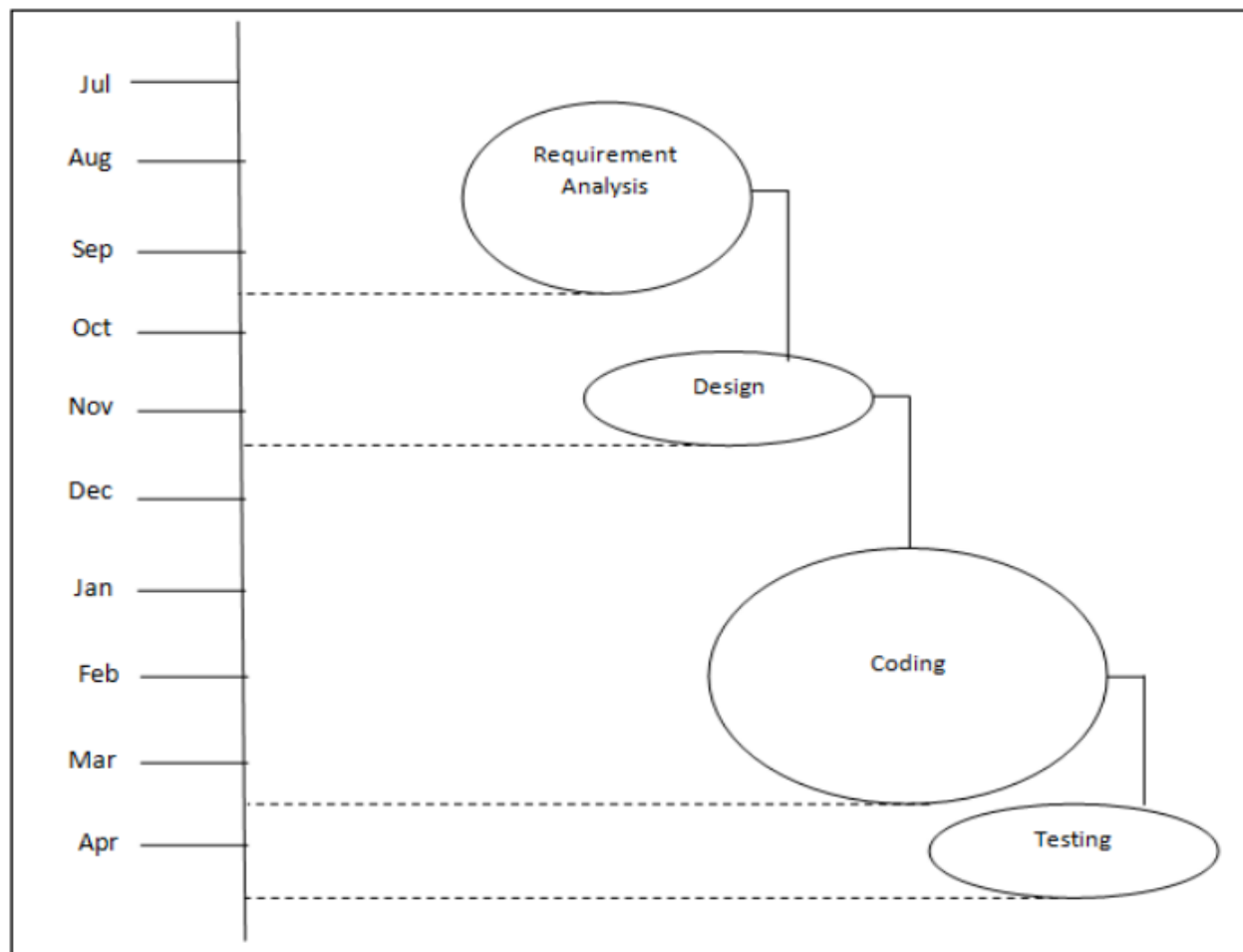
- To build a model to detect and ascertain the rumor on social media.
- To improve the accuracy of a proposed framework using hybrid deep learning model.
- To Classify a particular post of a user to desired outcome which involve(NON-RUMOR(NR) OR RUMOR(R))

## **1.4 SCOPE & LIMITATIONS**

- 1 To overcome the misled information which cause violence defamation hatred on the twitter social community.
- 2 The accurate acknowledgment of the incidence or events will be shared among the social community.

## 1.5 TIMELINE OF THE PROJECT

We started the project by gathering the related documents to the project at the end of July 2022. Gathering the requirements and all the analysis tasks was done by mid of August 2022. After that System design was started in the month of September 2022 and completed by the start of November along with the UML diagrams and Synopsis with a rough idea of the project. In November 2022 we started making the detailed SRS documents along with deciding the methodology for the project which was completed by mid-December 2022. By the start of January 2023, we started coding by dividing it into 5 modules and completed the 1st module by mid-January 2023. Other two modules were completed by the end of February 2023. By the end of March 2023 the other 2. We started testing the project alongside designing the GUI which was completed in the first week of April 2023.



## 1.6 PROJECT COST -

### ❖ Line of code

To develop the system lines of codes are required.

### ❖ KLOC

KLOC is the estimated size of the software product indicated in Kilo Lines of Code.

$$\begin{aligned} \text{KLOC} &= \text{LOC} / 1000 \\ &= 1363 / 1000 \\ &= 1.363 \end{aligned}$$

### ❖ Effort

The effort is only a function of the number of lines of code and some constants evaluated according to the different software systems.

$$\begin{aligned} E &= a ( \text{KLOC} )^b \\ &= 2.4 ( 1.363 )^{1.05} \\ &= 2.4 * 1.384 \\ &= 3.931 \end{aligned}$$

### ❖ Time

The amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, months.

$$\begin{aligned} \text{Time} &= c ( \text{Efforts} )^d \\ &= 2.5 ( 3.931 )^{0.38} \\ &= 2.5 * 1.68 \\ &= 4.205 \end{aligned}$$

### ❖ Persons Required

Persons required is nothing but effort divided by time.

$$\begin{aligned} \text{Persons Required} &= \text{Efforts} / \text{Time} \\ &= 3.931 / 4.205 \\ &= 0.93 \end{aligned}$$

## **2. BACKGROUND STUDY AND LITERATURE OVERVIEW**

- Al-Sarem M, Boulila W, Al-Harby M, Qadir J, Al Saeedi A (2019) Deep learning-based rumor detection on microblogging platforms: a systematic review. *IEEE Access* 7:152788–152812
- To Study a rumor detection system, to verify the truthfulness of posts uploaded by the user automatically by mining rich text available on the open network with deep learning techniques. Asghar MZ, Habib A, Habib A, Khan A, Ali R, Khattak A (2019) Exploring deep neural networks for rumor detection. *Journal of Ambient Intelligence and Humanized Computing*.
- Rumor detection System is a mechanism to verify the truthfulness of posts/tweets uploaded by the users across the globe. The widespread propagation of numerous rumors and fake news have seriously threatened the reliability of microblogs. They have investigated Rumor detection problem by exploring different Deep Learning models with emphasis on considering the contextual information in both directions: forward and backward, in a given text. Kotteti CMM, Dong X, Qian L (2020) Assemblage deep learning on time-series representation of tweets for rumor detection in social media. *Appl Sci* 10(21):7541
- The purpose of the Study is to design and develop rumor detection system in social media. They have proposed a data pre-processing method and ensemble model for fast detection of rumors on social media. The data pre-processing method transforms Twitter conversations into time-series vectors using the tweet creation timestamps, which can be extracted and processed without any wait time, which is a key requirement for the defined problem, and generated time-series data that are of the pure numeric type, which simplifies feature set complexity that in turn helps in reducing the computational complexity of classification models during their training process.
- An Efficient Fake News Identification System Using A-SQUARE CNN Algorithm pub Date:2022-02-19 pooja malhotra, sanjay kumar naik.
- This paper proposes a FastText Modified Embeddings from Language Models (FT-MELMo) based efficient FN identification system (FNIS). Specifically, the proposed system learns the semantic representation of data by utilizing FT-MELMo embedding.

- FT and MELMo utilize character n-grams and Bi-GRU, to equivalently maximize the classification accuracies and further minimize the training time. Lastly, a higher-dimensional embedding feature analogy is inputted to the AMSGradAdamNC based Convolutional Neural Network (A2CNN) classifier, which classifies the news as FN or real news effectively. The recommended A2CNN classifier accomplishes a classification accuracy of 96.54 percent with a low FDR of 4.32 percent. When in comparison with existing classifiers, the recommended A2CNN exhibits a low FPP and FNR, as reported by the performance comparison, the suggested FNIS is found to be exceptionally accurate, robust, and faster than other classifiers. An precise fake news detection approach based on a Levy flight honey badger optimized convolutional neural network model pub Date:2022 Dheeraj Kumar Dixit, Amit Bhagat, Dharmendra Dangi.. The spread of fake news has become easier as a result of the availability of these platforms. Anybody with access to these networks generates and distributes fake news for professional or personal gain. To Tackle these issues, This proposed approach initially obtains the data from the ISOT dataset and data cleaning is performed. After that, pre processing is done which includes three significant steps such as stemming, stop word removal, and tokenization. Next to pre processing, various features that associate name entity recognition-based features are selected during feature extraction. From this, the short extent features are selected with the help of the aggregate modified independent component analysis model. eventually, the hybrid convolutional neural network-based Levy flight-based honey badger algorithm detects fake news. The experiments are simulated using python software with diverse performance metrics such as accuracy, specificity, sensitivity, precision, and F-scores to validate the performance of the proposed method. The offered model suggest a precision, recall, and accuracy value of 95%, 97%, and 98% when estimate with the ISOT dataset. When compared to the existing state-of-art methods, the proposed method gave in superior detection results and higher accuracy rates.
- Effective fake news detection using graph and summarization techniques Pub Date:2021-08-10 Gihwan Kim, youngjoong Ko.
- In this Paper they propose a novel method that uses graph and summarization techniques for fake news detection. The proposed method represents the relationship of all sentences in a graph structure to accurately understand the context information of the document.

Accordingly, the relationship between sentences in the graph is calculated as a score through the attention mechanism. Then, the summarization technique is used to reflect the sentence subject information in the graph update process. The proposed method shows better performance than Karimi's and BERT based models by approximately 10.34%p and 3.72%p, respectively.

- Dynamic Graph Neural Network for Fake News Detection pub Date:2022-07-18 Chengaung song,siqi wei
- Fake news detection has become the research well focused in both academic and industrial communities. The most of existing wide distribution-based fake news detection algorithms are based on static networks and they assume the whole information wide distribution network structure is freely available before performing fake news detection algorithms.
- To address these shortcomings, This paper proposed a dynamic propagation graph-based fake news detection method to capture the missing dynamic propagation information in static networks and classify fake news. Specifically, the proposed method models each news generate graph as a series of graph snapshots recorded at distinct time steps. This paper evaluates the approach on three real-world benchmark datasets, and the experimental results demonstrate the effectiveness of the proposed model.

## **2.1 LITERATURE OVERVIEW**

In the previous rumor detection system we can observe that where more importance was given to text and temporal features and showed a moderate accuracy. Here we develop a system using BiLSTM with word embedding and MLP model with various features improves the accuracy.

## **2.2 CRITICAL APPRAISAL OF OTHER PEOPLE'S WORK**

### **A. Early Rumor Detection System by Kaimin Zhou.**

Early rumor detection learns dynamically the minimum number of posts required to identify a rumor, integrate reinforcement learning with recurrent neural networks to monitor social media posts in real time to decide when to classify rumor.

### **B. Analysis of Techniques for Rumor Detection in Social Media Ajeet Ram Pathak**

The study of various techniques and methodology for detecting rumors on social media, focuses on rumor detection, classification of the rumors and demystifies steps in rumor detection models. Datasets relate to rumor detection have also been thoroughly discussed highly developed rumor detection models based on machine learning, deep learning and hybrid approaches have been accurately analyzed and compared.

## **2.3 INVESTIGATION OF CURRENT PROJECT & RELATED WORK**

We Started this project with the aim of creating an efficient rumor detection system. The previous rumor detection system was using CNN approach to classify the rumors on social media by analyzing time series Which was less sufficient and time consuming So to overcome this limitation we decided to create a rumor detection system using post-wise essential features, Compared to the previous work, where more importance was given to text and temporal features and showed a moderate accuracy, in this project we have focused on on text, user, content-based, and lexical category features. The BiLSTM with word embedding and MLP model with various features improves the accuracy.



### **3. REQUIREMENT ANALYSIS**

#### **3.1 SOFTWARE AND HARDWARE REQUIREMENTS**

##### **3.1.1 Hardware Requirement:**

- Personal computer with minimum 2GB ram.
- 1TB hard disk.
- 64bit Operating System.

##### **3.1.2 Software Requirement:**

- Compatible with MAC, UBUNTU, WINDOWS.
- Python 3 64 bit.
- We can run code on Collab, Visual studio,pycharm, jupyter notebook.

### 3.2. REQUIREMENT SPECIFICATION (TABLE)

No.	Requirement	Essential / Desirable	Description of the Requirement	Remarks
RS1	For Detection purpose data should be Loaded.	Essential	Data should be Loaded of twitter.	---
RS2	Data should be properly preprocessed.	Essential	Data should be preprocessed.	---
RS3	The Model should be properly Define.	Essential	Model should be properly define.	---
RS4	Validation Strategy for Model should be define properly	Essential	Model validation strategy should be define.	---
RS5	Setting Correct Performance requirement metrics for Model.	Essential	Setting Correct performance requirement metrics.	---

Table 3.2 Requirement specification table

### 3.3 USE CASE DIAGRAM

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.



Fig 3.3 Use case diagram

## 4. SYSTEM DESIGN

### 4.1 ARCHITECTURE DIAGRAM

An architectural diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.

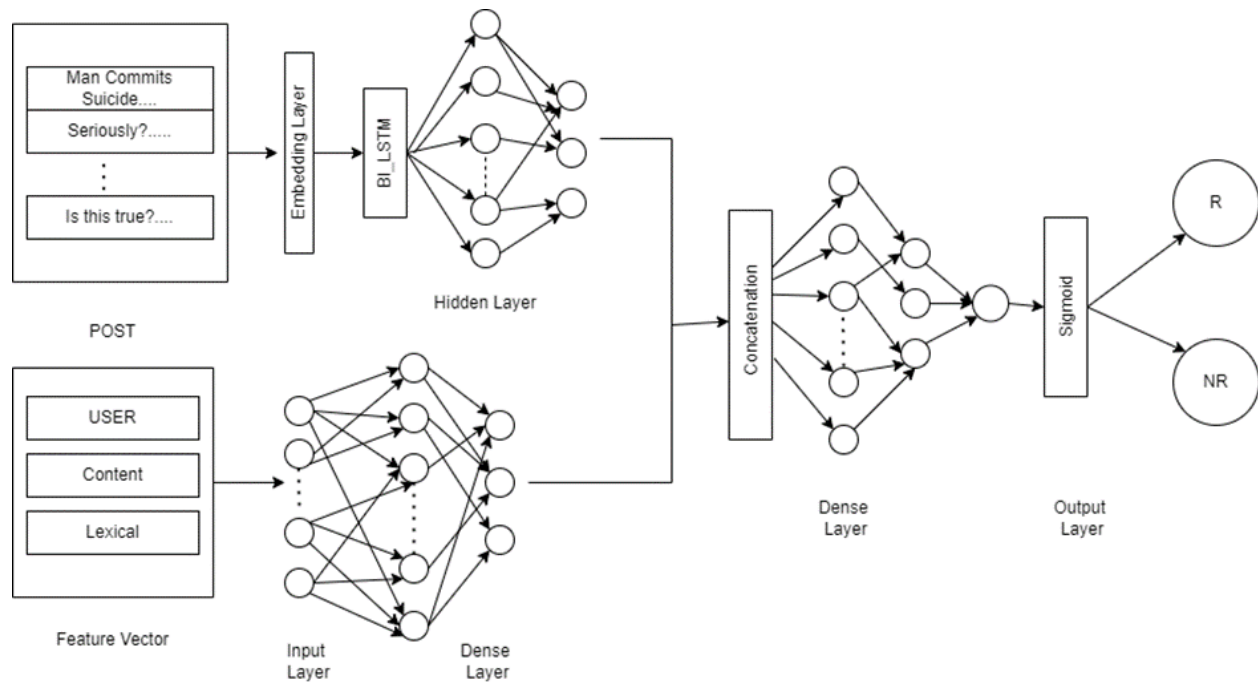


Fig 4.1 Architecture Diagram

### 4.2 MODULES

#### 4.2.1. Input Layer:

The input to the model is a sequence of tokens or words. Each token is typically represented as a vector using techniques like word embeddings or character embeddings.

INPUT - Sequence of tokens or words represented as vectors.

OUTPUT - Sequence of input vectors.

#### **4.2.2. BiLSTM Layer (Hidden Layer):**

The input sequence is passed through a bidirectional LSTM layer. An LSTM (long short-term memory) is a type of recurrent neural network (RNN) that is capable of learning long-term dependencies in sequential data. The bidirectional aspect means that the input sequence is processed in both forward and backward directions, allowing the model to capture contextual information from both past and future tokens.

INPUT - Sequence of input vectors.

OUTPUT - Bidirectional LSTM-encoded sequence.

#### **4.2.3. UCL Layer (Dense Layer):**

The output of the BiLSTM layer is fed into the UCL, which is responsible for unsupervised clustering of the encoded representations. The UCL aims to group similar representations together without the need for explicit labels or supervision. One common technique used in the UCL layer is K-means clustering, where the representations are clustered into K groups based on their similarity.

INPUT - Bidirectional LSTM-encoded sequence

OUTPUT - Clustering assignments or centroids.

#### **4.2.4. Output Layer:**

The final output of the model depends on the specific task. For example, in sequence labeling tasks such as named entity recognition or part-of-speech tagging, each token in the input sequence may be assigned a label. In text classification tasks, the model may predict a single label or multiple labels for the input sequence.

INPUT - Clustering assignments or centroids.

OUTPUT - Task-specific predictions.

## **4.3 ALGORITHMIC DESCRIPTION OF EACH MODULE**

### **4.3.1. Input Layer:**

This module is the very first module to be implemented, this module accepts sequence of tokens or words.

Step 1 - Begin.

Step 2 - Enter sequence of tokens or words.

Step 3 - End.

### **4.3.2. BILSTM Layer (Hidden Layer):**

Step 1 - Begin.

Step 2 - Initialize forward and backward LSTM hidden states.

Iterate over the input sequence.

Step 3 - Compute the forward LSTM hidden state based on the current input

Vector and t previous forward hidden state

Step 4 - Compute the backward LSTM hidden state based on the current input

vector and the previous backward hidden state

Step 5 - Concatenate the forward and backward hidden states to obtain the final

encoded representation for the current input vector.

Step 6 - End.

### **4.3.3. UCL Layer (Dense Layer):**

Step 1 – Begin

Step 2 - Perform unsupervised clustering (e.g., K-means clustering) on the encoded sequence to group similar representations together.

Step 3 - Obtain the clustering assignments or centroids based on the

chosen clustering algorithm

Step 4 - End.

#### **4.3.4. Output Layer:**

Step 1 – Begin

Step 2 - Sequence Labeling:

Iterate over the encoded sequence:

1. Assign a label to each token based on the clustering assignments or centroid
2. Output the sequence of labels.

Step 3 - Text Classification:

- Feed the clustering assignments or centroids into a classification layer (e.g., fully connected layers).
- Apply appropriate activation functions (e.g., SoftMax) to obtain the probability distribution over different classes.
- Output the predicted class or classes.

Step 4 - End.

## 4.4 SYSTEM MODELLING

### 4.4.1 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

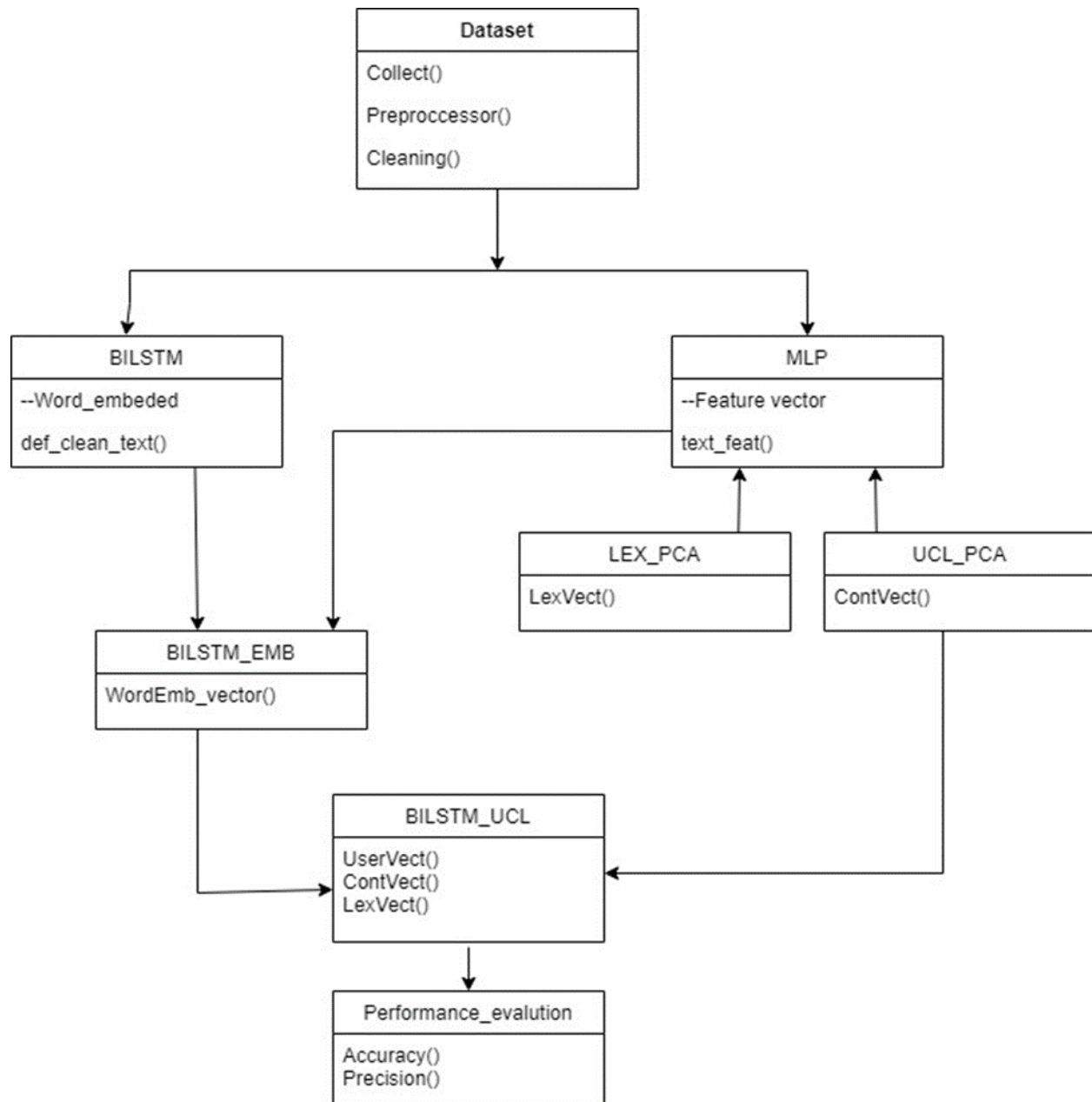


Fig 4.4.1 Class diagram



#### 4.4.2 Data flow diagram

A DFD is a graphical representation of flow of data through an information system.

- **DFD level 0: -**

DFD level 0 shows the flow between the prominent members of the project.

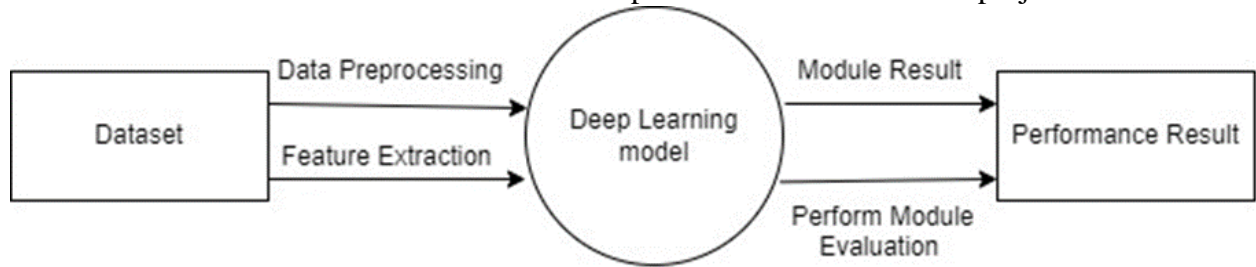


Fig 4.4.2a DFD level 0

- **DFD level 1: -**

DFD level 1 consists of the sub-members along with the main members required for the flow of information

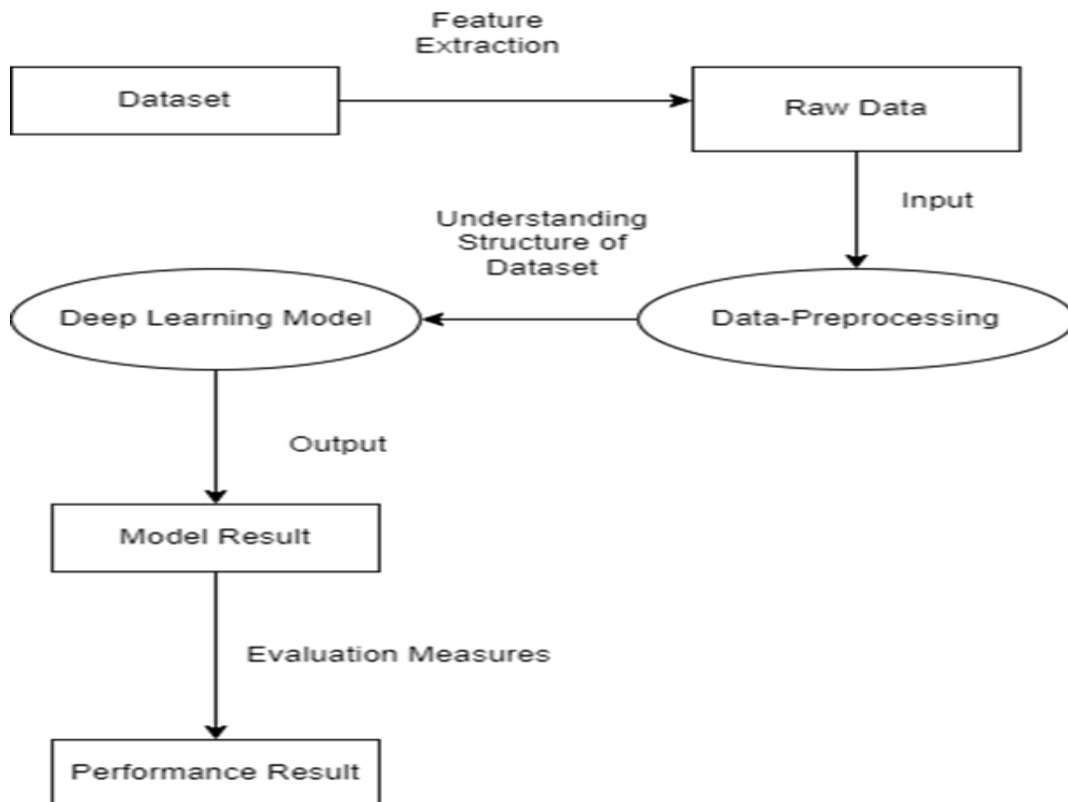


Fig 4.4.2b DFD level 1

- **DFD level 2: -**

This is the most detailed level of DFDs, which provides a detailed view of the processes, data flows, and data stores in the system. This level is typically used for complex systems, where a high level of detail is required to understand the system.

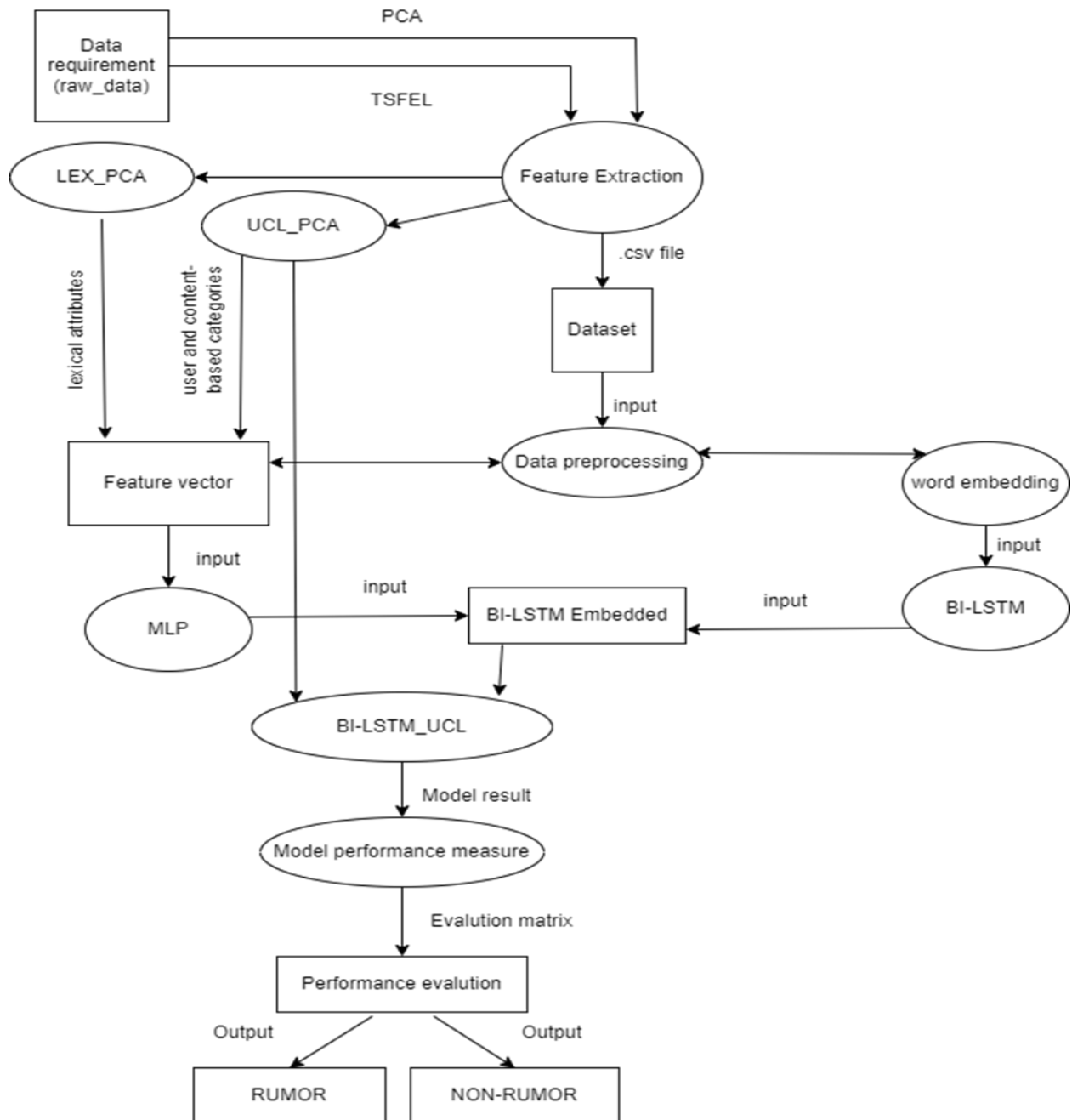


Fig 4.4.2c DFD level 2

## 4.5 ACTIVITY DIAGRAM

- Activity diagrams are also called object-oriented flowcharts diagram.
- Flowcharts diagrams consist of activities that are made up of smaller actions.
- Activity is a action that is divided into one or more actions.

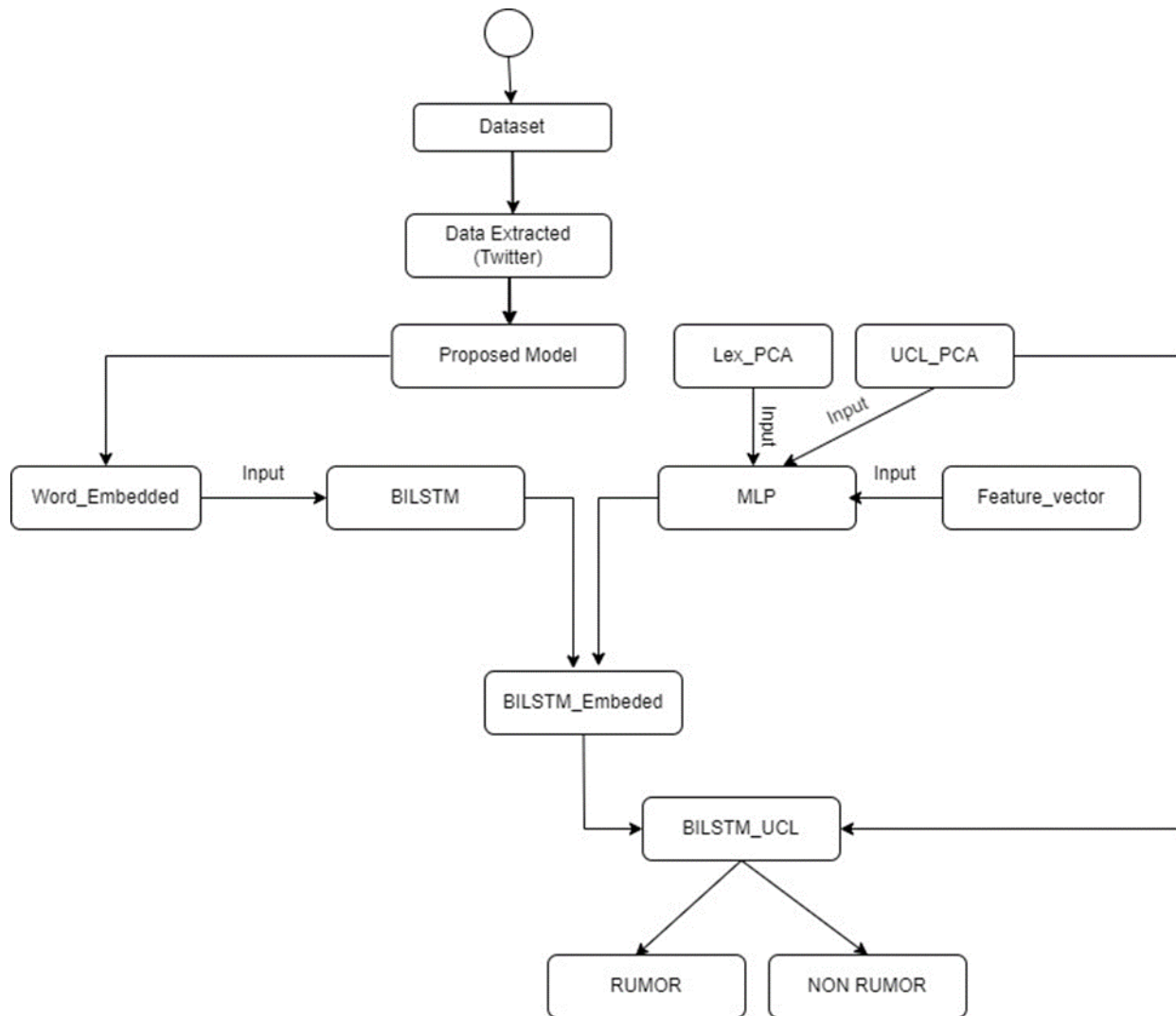


Fig 4.5 Activity Diagram

## 5 IMPLEMENTATION

### 5.1 ENVIRONMENT SETTING FOR RUNNING THE PROJECT

Sr no.	Technology Specifications
1	Operating System Windows 7 and Higher.
2	Python 3.6 or higher, Tensor Flow, Keras and other Relevant python packages
3	IDE Vs studio.
4	Goggle collab

### 5.2 DETAILED DESCRIPTION OF METHODS

Rumor detection on Twitter dataset using BiLSTM and UCL model involves the following methods:

#### 5.2.1 Data Preprocessing:

Data preprocessing is the process of cleansing, arranging, and transforming the data into a format suitable for analysis. This step involves as tokenization, stop-word removal, stemming, and other text processing techniques. In this method, the authors used pre-processing techniques such as lowercasing the text, replacing URLs, removing punctuation, and removing stop words. The data pulled from Twitter/ social media contains URLs, hashtags, mentions, emoticons, special characters, etc., which need to be preprocessed to use the cleaned text data as input to our model. The text data is get ready by removing URLs, hashtags, mentions, emoticons, punctuations, and non-ASCII characters using the regex library which is represented by re. The duplicate and same

posts are removed from the dataset. These cleaned posts are used to retire lexical features and word embedding. The content-based components are pulled from the original tweets.

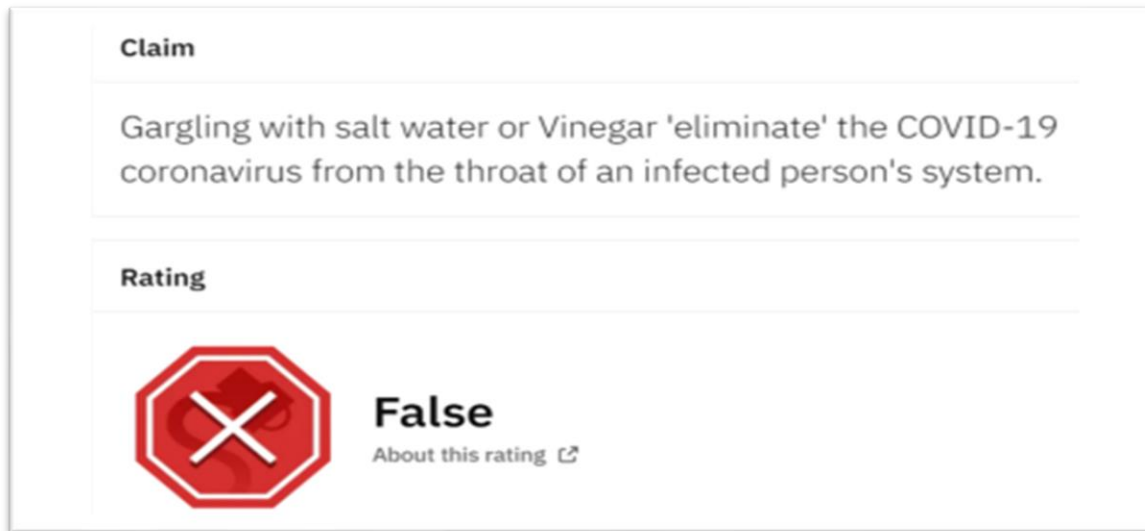


Fig 5.2.1 Identification of event as rumour

Data encoding in the numerical form must be necessary for the text data to input a deep neural network. Besides the comparative average length of all posts, we allow the maximum lengths for the sequence as 100. The Tokenizer divided the post into different tokens and utilized as a word dictionary. This word dictionary is exploited to transform the message into a sequence of integers applying the text\_to\_sequence function. filling after the post is performed to create all sequences equal to maximum length. The embedding layer is used to appreciate the meaning of words in a post, transforming each word into an n-dimensional word embedding vector by having a sequence of posts as an input data. The output data of this embedding layer transferred as an input to the BiLSTM model.

#### 5.2.1.1 Word embedding

The dataset contains several tweets and every tweet is encompassed in an order of n words, i.e.  $t_1, t_2, \dots, t_n$ . Each word transforms into an embedding vector  $w_i \in E_m$ , called as word embedding. The Keras embedding layer is used in this research. The input to the embedding layer holds an input matrix of two dimensional, also called as word embedding matrix which is

represented by  $Elxm$ , where  $l$  is the tweet's length, and  $m$  is represent the dimension of word embedding. The variable used in the Embedding layer are the size of embedding dimension ( $m$ ) = 32, vocabulary size = 1000 and length of the sequence ( $l$ ) = 100.

#### **5.2.1.2 Pseudocode for Text processing**

```
def clean_text(post):
    clean_pst = re.sub('http\S+\s', x) # remove URLs
    clean_pst = re.sub('RT, clean_pst) # remove RT clean_pst-re.sub('#S+' clean_pst) # remove
    hashtags
    clean_pst-re.sub('@S+ ', clean_pst) # remove mentions
    clean_pst = Expand contractions
    clean_pst = Remove punctuation symbols clean_pst Remove emoticons, symbols and pictographs
    clean_pst = Remove non ascii characters
    clean_pst = clean_pst lower() #Convert text to lower case clean_pst-clean_pst.strip()
    return clean_pst.
```

#### **5.2.2 Feature Extraction**

Feature extraction involves extracting relevant features from the preprocessed data. In this method, the authors used word embeddings to represent the text. Word embeddings are dense vector depiction of words that capture their semantic meaning. The authors used pre-trained word embeddings such as fast Text.

#### **5.2.3 BiLSTM Model**

The BiLSTM model is a deep learning model that is effective in handling sequential data. The model consists of a bidirectional LSTM layer followed by a dense layer. The BiLSTM layer processes the input sequence in both forward and backward directions and captures the contextual information of the sequence. The dense layer takes the output of the BiLSTM layer and generates the final output. In this method, the authors used a BiLSTM model with two hidden layers.

#### 5.2.4 **UCL Model**

The UCL model is a rumor detection model that uses a combination of features such as content, network, and temporal features. The UCL model consists of multiple layers, including a content layer, a network layer, and a temporal layer. The content layer processes the textual information, the network layer processes the social network information, and the temporal layer processes the time-related features. In this method, the authors used the UCL model to extract the features from the Twitter dataset.

#### 5.2.5 **RNN based approach:**

RNN is a form of a feed-forward neural network used to process the sequential data with a variable-length, such as time-series data and is the first to apply for rumor detection by Ma et al. They extended the basic RNN model with memory unit models like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), where GRU performs well.

The words related to rumor should get more attention, as proposed by Chen et al. using multilayer LSTM and deep attention models. The model depicts the soft attention to the recurrence of distinct features with a specific focus and produces hidden representations in the posts. Guo et al. proposed the same attention-based technique for the hierarchical network of word-post-subevent using bidirectional LSTM. They utilized propagation features such as an average of reposts and comments along with user and post-based features. Chen et al. proposed an unsupervised learning model for rumor detection as an anomaly detection problem on Sina Weibo, a Chinese microblogging website.

They utilized microblogs features (like Question mark count, Sentiment score, Pictures count) and posts (like length, count of likes, URL count). The experimental results show that their projected method could attain an accuracy of 92% and an F1 score of 89%. The self-learning semi-supervised deep learning model and the trust network layer are used in the Fake News Net dataset, which uses a bidirectional LSTM (BiLSTM) model with a trust layer and shows an F1 score of 88%.

#### 5.2.6 **Hybrid Models:**

Ruchan sky et al. proposed a model for fake news detection based on the text of an article, the temporal activity of user response and source users propagating it. They put forward a hybrid

model by integrating features from all three categories to get a more precise rumor classification. A recommender system determines a user's genuine interest based on user involvement therefore, user characteristics are vital. Song et al proposed a CNN-based model for credible early detection of rumor where they extract feature vectors in each interval using CNN and feed it to RNN. Also, they suggest that other features related to the user profile and propagation patterns can improve rumor detection. Liu et al proposed a model for early rumor detection, which combines RNN and CNN to capture temporal patterns of global and local features of users with the propagation paths. They utilized user-related characteristics such as user registration age, geo-enabled, verified users, etc.

Data based on information campaigns and promoting is used and proposed a Generative Adversarial Network network (GAN) for rumor detection by Ma et al. Kumar et al proposed a model in which a sentiment analysis of social 17350 Multimedia Tools and Applications (2022) 81:17347–17368 network users utilized various deep learning models like CNN, a variation of RNN as long short-term memories (LSTM) with ensemble and attention mechanism. They used Glove (global vector for word representation) for word embedding.

Other than the proposed categories of deep learning-based approaches, the researchers have used ensemble learning for rumor detection and transfer learning for fake news classification. An ensemble learning combination of RNN, GRU and LSTM models is used with various layers in the neural network by Kotteti et al.

A transfer learning using BERT (Bidirectional Encoder Representations from Transformers) model referred to as Fake BERT uses a combination of deep Convolutional Neural Network (CNN) with different kernel sizes and filters by Kaliyar et al.

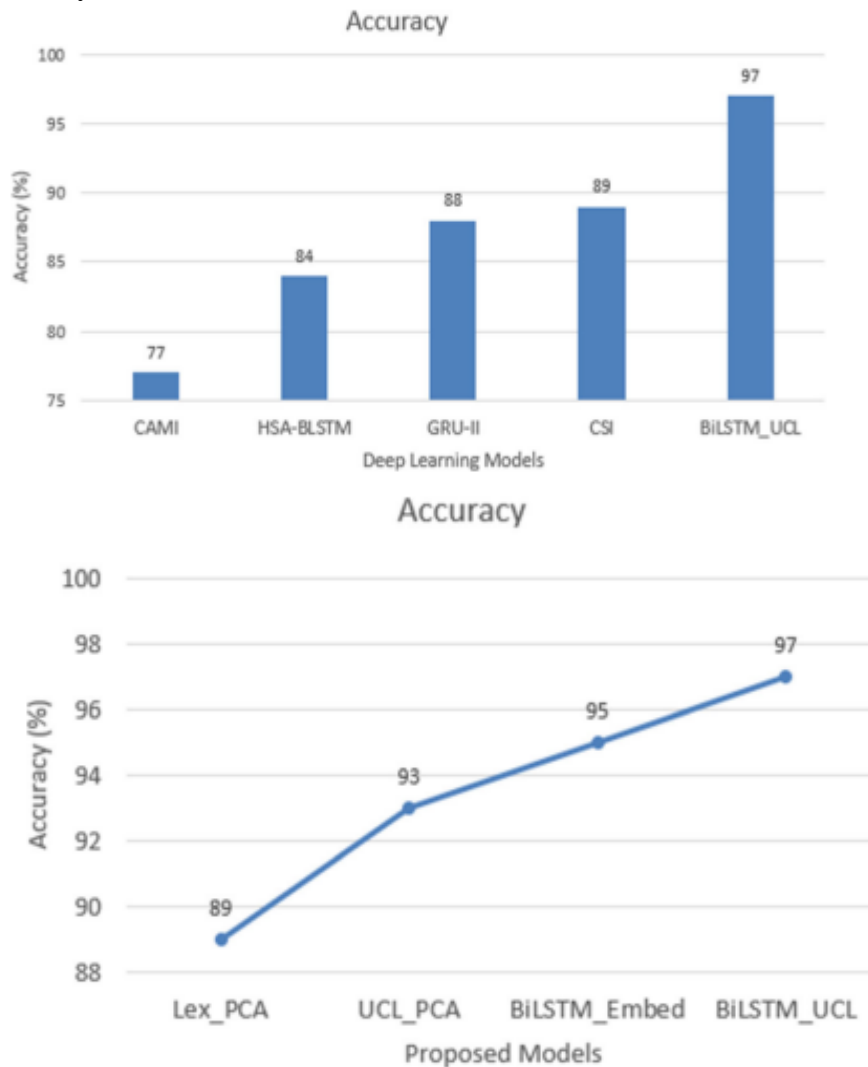
### 5.2.7 **Training and Evaluation:**

The model was trained using the annotated dataset and evaluated using performance metrics such as precision, recall, F1-score, accuracy, and confusion matrix. The model was trained using a training set and evaluated using a validation set. The authors used a 10-fold cross-validation method to evaluate the performance of the model.



### 5.2.8 Comparison with Other Methods:

The performance of the BiLSTM and UCL models were compared with other state-of-the-art rumor detection methods such as SVM, Naive Bayes, and Random Forest. The results showed that the BiLSTM and UCL models outperformed the other methods in terms of accuracy and F1-score.



Overall, the BiLSTM and UCL models are effective in rumor detection on Twitter dataset due to their ability to handle sequential data and extract relevant features from the dataset. The methods used in this study can serve as a guide for researchers and practitioners in developing more effective rumor detection models.

### **5.3 IMPLEMENTATION DETAILS**

The BiLSTM\_UCL model in rumor detection combines a Bidirectional Long Short-Term Memory (BiLSTM) network with additional user, content, and location (UCL) features. Here are the implementation details for the BiLSTM\_UCL model in rumor detection:

#### **5.3.1. Data Preprocessing:**

- Tokenization: The text data (tweets) is tokenized, splitting them into individual words or tokens.
- Padding: The tokenized sequences are padded or truncated to a fixed length to ensure uniformity in input size.

#### **5.3.2. Embedding Layer:**

- Initialize an embedding layer with pre-trained word embeddings such as Word2Vec or GloVe.
- Map the tokenized sequences to their corresponding word embeddings.

#### **5.3.3. BiLSTM Layer:**

- Add a Bidirectional LSTM layer to capture both the forward and backward contextual information.
- Set the number of hidden units and other hyperparameters for the LSTM layer.
- Connect the embedding layer to the BiLSTM layer.

#### **5.3.4. UCL Features:**

- Define additional input features such as user credibility, content sentiment, and location.
- Concatenate these features with the output of the BiLSTM layer.

#### **5.3.5. Classification Layer:**

- Add a fully connected layer or multiple dense layers on top of the concatenated features.
- Set the appropriate activation function for the output layer based on the classification task (e.g., sigmoid for binary classification).
- Train the model using appropriate loss function (e.g., binary cross-entropy) and optimizer (e.g., Adam).

#### **5.3.6. Model Training:**

- Split the dataset into training, validation, and testing sets.
- Compile the model with the chosen loss function, optimizer, and evaluation metrics.
- Train the model using the training data, validating the performance on the validation set.
- Adjust the hyperparameters (e.g., learning rate, batch size, number of epochs) based on the validation results.

#### **5.3.7. Evaluation:**

- Evaluate the trained model on the testing set using metrics such as accuracy, precision, recall and F1-score.
- Analyze the models performance and compare it with other models or baselines.

#### **5.3.8. Deployment:**

- Save the trained model to disk for future use.
- Build an API or user interface to interact with the model and make predictions on new data.

## **5.4 Code for Bi-LSTM UCL**

```
from keras.layers import RepeatVector

from keras.layers import Flatten

embedding_dim = 100 # Define the dimensionality of the word embeddings

vocab_size = len(tokenizer.word_index) + 1

inputs = Input(shape=(max_length,))

embedding_layer = Embedding(vocab_size, embedding_dim, input_length=max_length)(inputs)

lstm_layer = Bidirectional(LSTM(64, dropout=0.2, recurrent_dropout=0.2))(embedding_layer)

dropout_layer = Dropout(0.2)(lstm_layer)

auxiliary_output = Dense(1, activation='sigmoid', name='aux_output')(dropout_layer)

reshaped_lstm_layer = RepeatVector(max_length)(lstm_layer)

flatten_embedding_layer = Flatten()(embedding_layer)

reshaped_embedding_layer = RepeatVector(max_length)(flatten_embedding_layer)

merged_layer = concatenate([reshaped_embedding_layer, reshaped_lstm_layer])

dense_layer = Dense(64, activation='relu')(merged_layer)

dropout_layer2 = Dropout(0.2)(dense_layer)

output = Dense(1, activation='sigmoid', name='main_output')(dropout_layer2)

model = Model(inputs=inputs, outputs=[auxiliary_output, output])

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'],

loss_weights=[0.2, 1.0])
```

## 6.INTEGRATION AND TESTING

### 6.1 UNIT TESTING

Test cases	Expected output	Actual output	Pass/fail
Test dataset	No Duplication in dataset.	No Duplication in dataset	Pass
Dataset with Null Value	Null value is present	Null value is present.	Pass
Upload of dataset	Extension of uploaded dataset file should be .csv	Extension of uploaded dataset file should be .csv	Pass
Extra Space	Extra space should not be present in data	Extra space is not present in dataset	Pass
Special Character in dataset	Special Character should not be present in dataset	Special Character is present in dataset	Pass

Table 6.1 Unit testing

## 6.2 INTEGRATION TESTING

Test cases	Expected output	Actual output	pass/fail
Verify if the dataset is properly uploaded.	Data uploaded successfully.	Data uploaded Successfully.	Pass
Verify if the data Tokenization is done properly.	Tokenization performed successfully.	Tokenization performed successfully.	Pass
Verify if the dataset is properly split into training and testing sets	Dataset successfully split into train and test set.	Dataset successfully split into train and test set.	Pass
Verify if the model is build properly.	Model build successful.	Model build successful.	Pass
Verify if the prediction of model is correct.	Prediction of model is correct.	Prediction of model is correct.	Pass

Table 6.2 Integration testing

## 7 PERFORMANCE ANALYSIS

For evaluation metrics, we adopted Accuracy, Precision, Recall and F1 scores for a comprehensive evaluation. confusion matrix summarizes predicated results over actual results, where R stands for rumor and NR for non rumor. Accuracy is a fraction of correct predictions overall predictions. The quantity of accurate positive results divided by the quantity of positive results predicted by the classifier is called Precision. The recall is the quantity of correct positive results divided by the amount of all relevant samples.

- Accuracy = 
$$\frac{TP + TN}{TP + TN + FP + FN}$$
- Precision = 
$$\frac{TP}{TP + FP}$$
- Recall = 
$$\frac{TP}{(TP + FN)}$$
- F1 = 
$$\frac{2 * Recall * Precision}{Recall + Precision}$$

		Predicated Class	
		Positive(R)	Negative NR)
Actual Class	Positive(R)	True Positive (TP)	False Negative (FN)
	Negative NR)	False Positive (FP)	True Negative (TN)

Table 7. Confusion Matrix

## **8 FUTURE SCOPE**

Rumor detection on Twitter datasets has a promising future scope due to the increasing reliance on social media platforms for information dissemination and the challenges associated with the spread of misinformation. Here are some potential areas of future development and research in rumor detection on Twitter datasets:

### **8.1 Advanced Machine Learning Techniques:**

Researchers can explore the use of advanced machine learning algorithms, such as deep learning models, to improve the accuracy and efficiency of rumor detection. Techniques like recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformers can be employed to capture complex patterns in textual data and detect rumors effectively.

### **8.2. Real-time Detection:**

Developing real-time rumor detection systems will be crucial to counter the rapid spread of rumors on social media platforms. These systems would continuously monitor Twitter feeds, analyze incoming tweets, and quickly identify potential rumors for timely intervention. Integration with natural language processing (NLP) techniques and efficient data processing strategies would be essential to achieve real-time detection.

### **8.3. Multi-modal Analysis:**

Twitter data comprises not only textual information but also images, videos, and other forms of multimedia content. Integrating multi-modal analysis techniques, which combine text, visual, and audio features, can enhance the accuracy of rumor detection. Deep learning architectures, such as multimodal transformers, can be employed to process and extract information from different modalities simultaneously.

### **8.4. User Credibility and Influence:**

Evaluating user credibility and influence is crucial in rumor detection. Future research can focus on developing methods to identify influential users or "super spreaders" of rumors on Twitter. Techniques such as social network analysis, sentiment analysis, and user behavior



modeling can be employed to understand user dynamics and assign credibility scores to users, aiding in rumor verification.

### **8.5 Contextual Information:**

Incorporating contextual information, such as temporal and spatial factors, can improve the accuracy of rumor detection. Analyzing the temporal dynamics of tweet propagation and considering geographical information can provide additional cues for identifying rumors. Additionally, considering the broader context, such as ongoing events or trending topics, can help differentiate rumors from legitimate information.

### **8.6. Explainable AI:**

Developing explainable AI models for rumor detection is essential to gain user trust and enhance transparency. By providing explanations for the predictions made by the models, users can better understand the rationale behind rumor detection outcomes. This could involve techniques such as attention mechanisms and interpretability methods to highlight relevant features and provide meaningful explanations.

### **8.7. Dataset Creation:**

Creating large-scale and diverse labeled datasets specifically tailored for rumor detection on Twitter would greatly benefit future research. These datasets should cover various domains, languages, and types of rumors to ensure the robustness and generalizability of the developed models.

### **8.8. Collaborative Efforts:**

Collaboration between researchers, social media platforms, and fact-checking organizations is crucial to address the challenges of rumor detection effectively. Sharing datasets, insights, and methodologies can lead to the development of more powerful and accurate rumor detection systems. Overall, the future of rumor detection on Twitter datasets lies in the integration of advanced machine learning techniques, multi-modal analysis, user credibility assessment, contextual information, explainable AI, and collaborative efforts. By combining these approaches, we can expect significant advancements in the field, leading to improved accuracy .

## **9 APPLICATIONS**

Rumor detection can have various applications in different fields. Some of the most common applications of rumor detection include:

### **9.1 Social media analysis:**

With the increasing use of social media, rumor detection can help identify false information and prevent the spread of rumors on social media platforms. Social media platforms such as Facebook and Twitter have integrated rumor detection systems to prevent the spread of misinformation.

### **9.2 Disaster response:**

In times of natural disasters or emergencies, false information can spread rapidly, causing chaos and panic. Rumor detection can help identify false information and prevent the spread of rumors, thereby improving the efficiency of disaster response.

### **9.3 Political campaigns:**

Rumors and false information can be spread during political campaigns to influence the outcome of the election. Rumor detection can help identify false information and prevent its spread, promoting transparency and fairness in political campaigns.

### **9.4 Business reputation management:**

Rumors can damage the reputation of businesses, affecting their sales and profitability. Rumor detection can help businesses identify false information and take appropriate measures to address the issue and protect their reputation.

### **9.5 News media:**

Rumors can spread quickly in the news media, leading to the dissemination of false information. Rumor detection can help journalists identify false information and prevent the spread of rumors in the news media.

## 10 OUTPUTS

```

twitter (1).ipynb X
G:\project> twitter (1).ipynb > df.head()
+ Code + Markdown + Outline ...
    if pred < 0.5:
        predicted_y.append(0)
    if pred >= 0.5:
        predicted_y.append(1)
    print(predicted_y[1])

[57]
... 0

confusion_matrix(y_test,predicted_y)

[58]
... array([[3103, 316],
          [ 230, 2386]])

from sklearn.metrics import accuracy_score
accuracy_score(y_test,predicted_y)

[59]
... 0.9095277547638774
    
```

Fig 1 Accuracy Of BiLSTM Model

```

+ Code + Text
0s array([[3101, 318],
         [ 246, 2370]])

[37] from sklearn.metrics import accuracy_score
pred=accuracy_score(y_test,predicted_y)

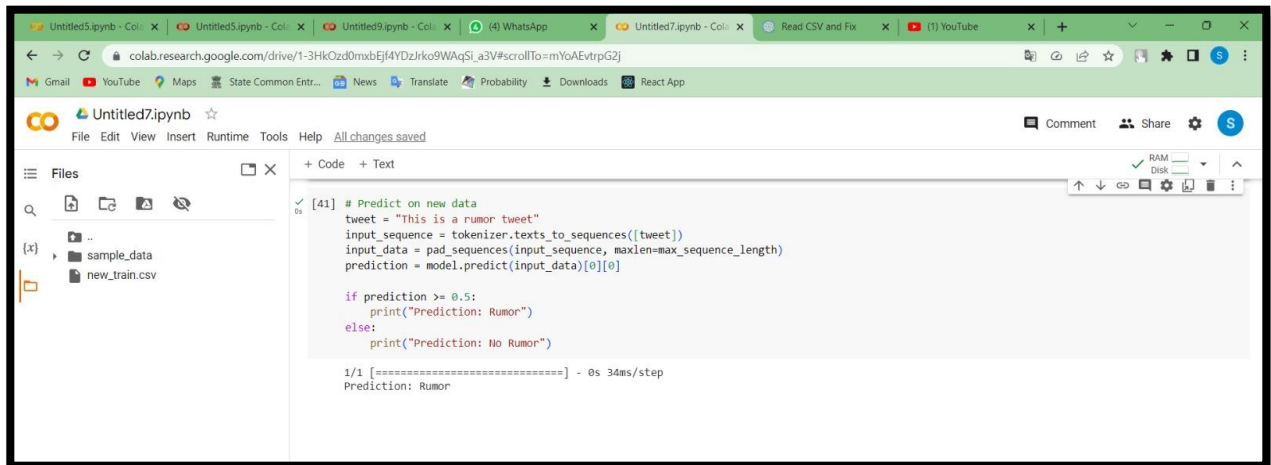
[38] print(pred)

0.9065451532725767

0s if pred >= 70:
    print("prediction model accuracy:Non rumor")
else:
    print("prediction model accuracy:Rumor")

prediction model accuracy:Rumor
    
```

Fig 2 Prediction with Accuracy Of BiLSTM Model



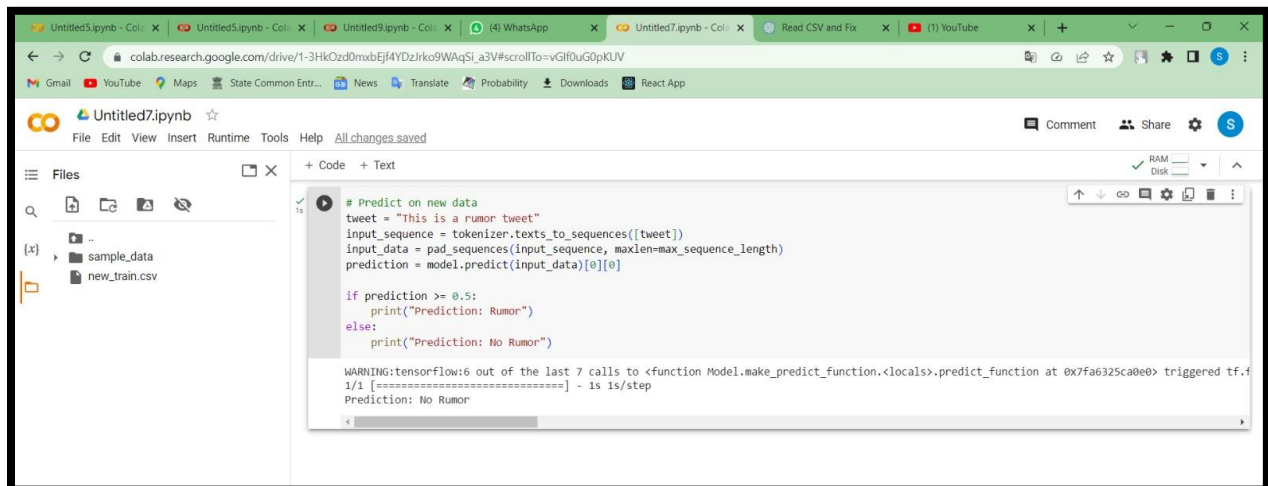
The screenshot shows a Jupyter Notebook interface with a file explorer on the left containing 'sample\_data' and 'new\_train.csv'. The code cell contains the following Python code:

```
[41] # Predict on new data
tweet = "This is a rumor tweet"
input_sequence = tokenizer.texts_to_sequences([tweet])
input_data = pad_sequences(input_sequence, maxlen=max_sequence_length)
prediction = model.predict(input_data)[0][0]

if prediction >= 0.5:
    print("Prediction: Rumor")
else:
    print("Prediction: No Rumor")

1/1 [=====] - 0s 34ms/step
Prediction: Rumor
```

Fig 3 Prediction -Rumor ( BiLSTM\_UCL Model)



The screenshot shows a Jupyter Notebook interface with a file explorer on the left containing 'sample\_data' and 'new\_train.csv'. The code cell contains the following Python code:

```
# Predict on new data
tweet = "This is a rumor tweet"
input_sequence = tokenizer.texts_to_sequences([tweet])
input_data = pad_sequences(input_sequence, maxlen=max_sequence_length)
prediction = model.predict(input_data)[0][0]

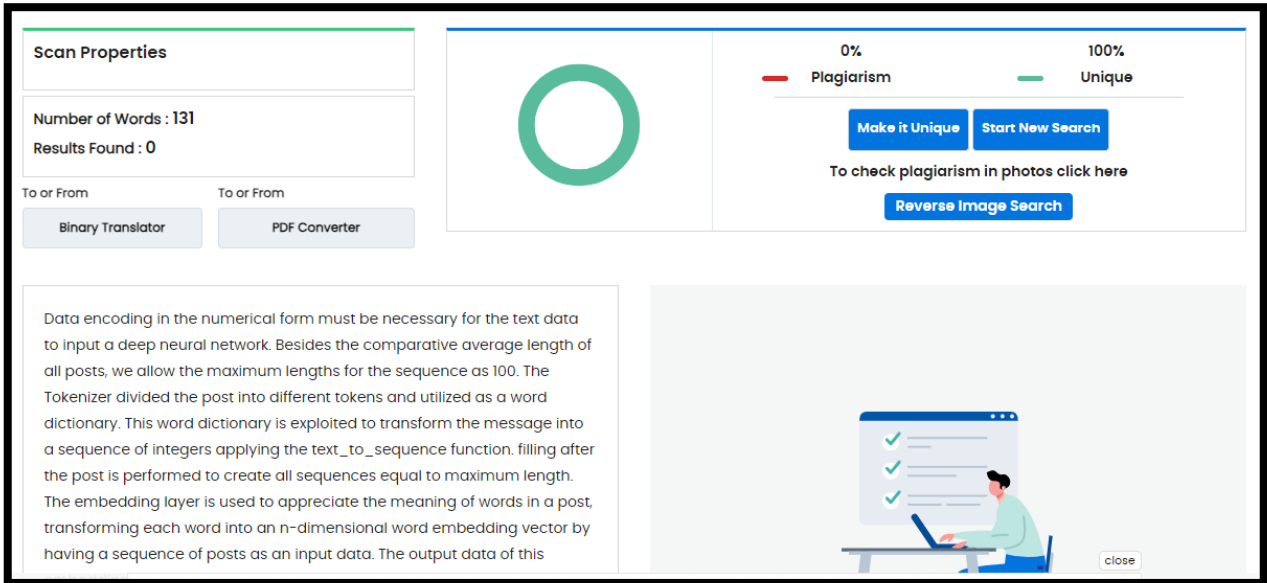
if prediction >= 0.5:
    print("Prediction: Rumor")
else:
    print("Prediction: No Rumor")
```

Below the code, a warning message is displayed:

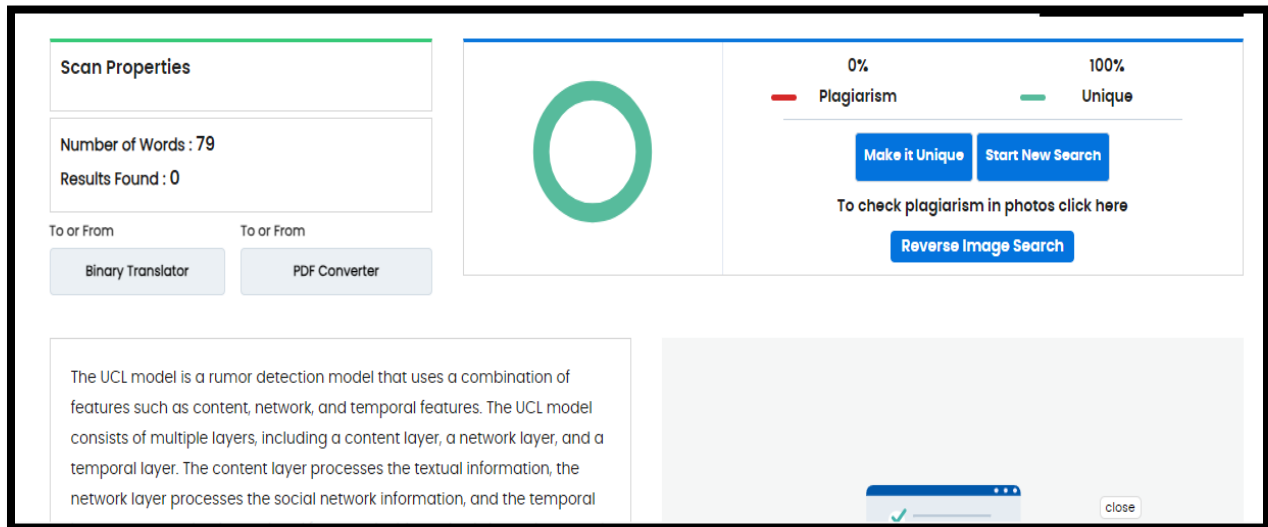
```
WARNING:tensorflow:6 out of the last 7 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7fa6325ca0e0> triggered tf.f
1/1 [=====] - 1s 1s/step
Prediction: No Rumor
```

Fig 4 Prediction -Non-Rumor ( BiLSTM\_UCL Model)

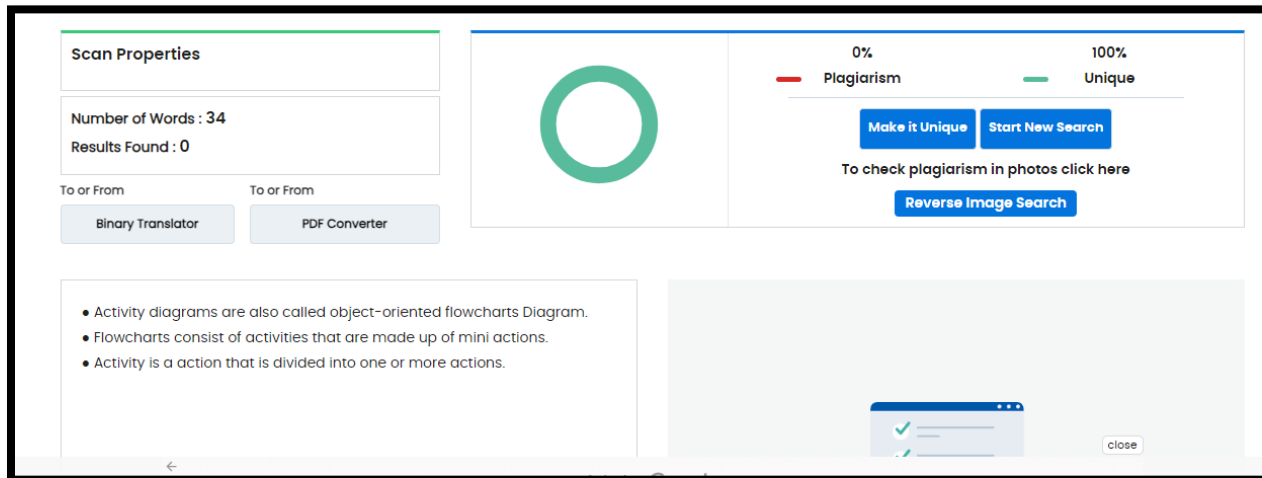
## 11 PLAGIARISM REPORT



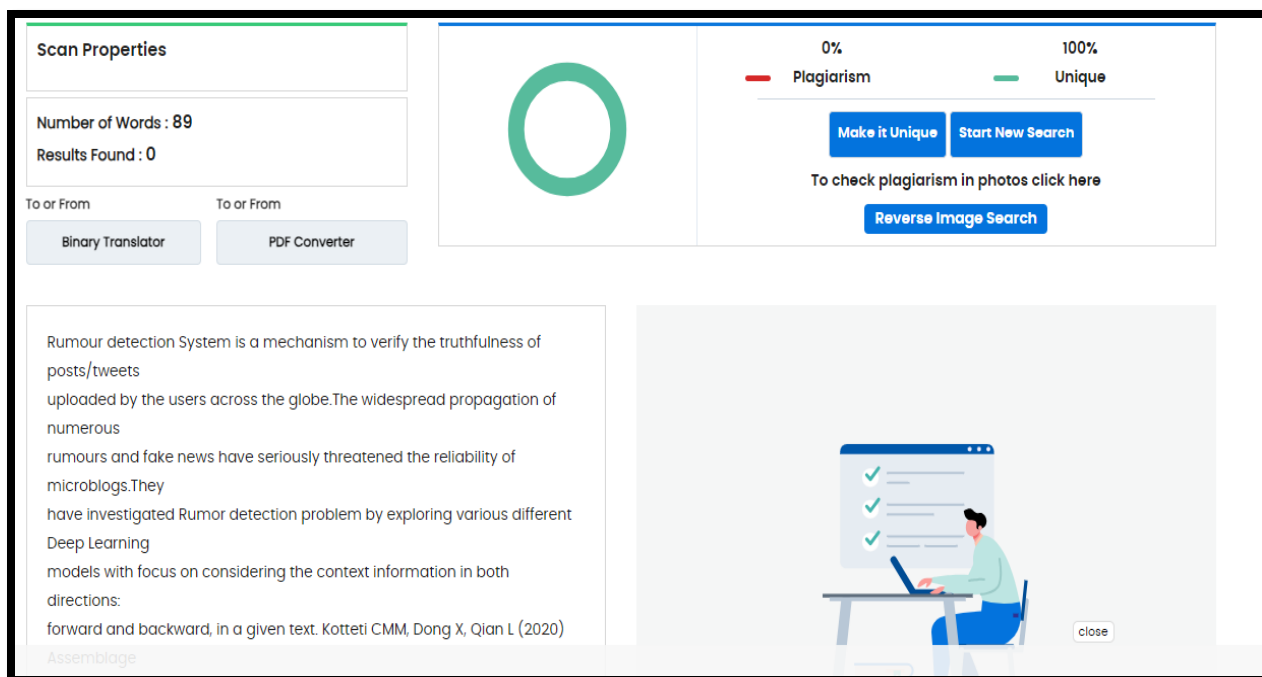
Snapshot 1



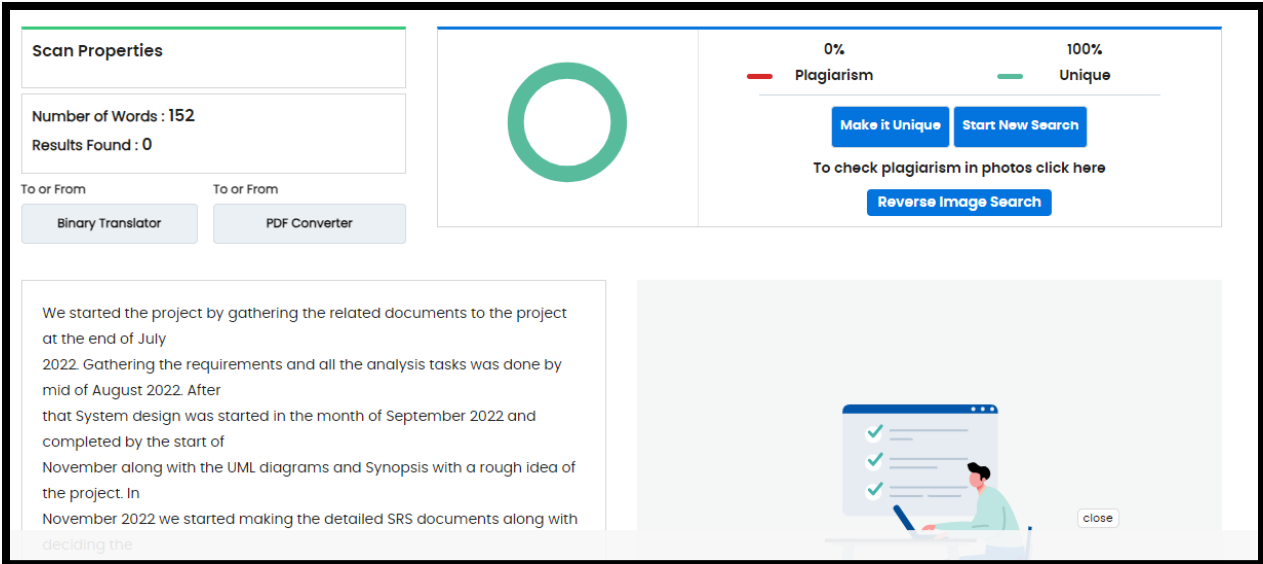
Snapshot 2



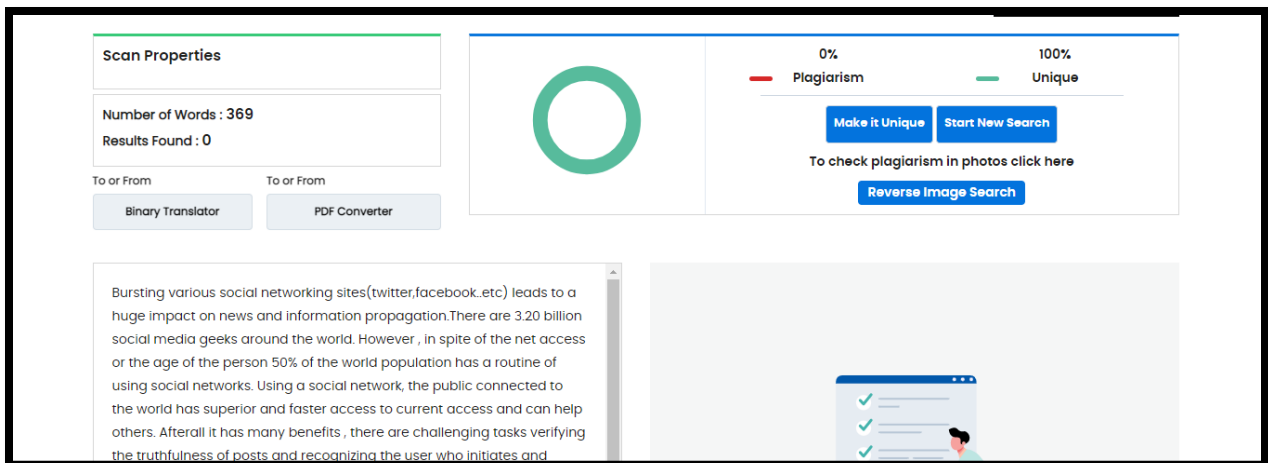
Snapshot 3



Snapshot 4



Snapshot 5



Snapshot 6

## 12 REFERENCES

- Shelke, S., Attar, V. Rumor detection in social networks based on user, content and lexical features. *Multimed Tools Appl* 81, 17347–17368 (2022). <https://doi.org/10.1007/s11042-022-12761-y>
- Bisong, E., 2019. Google colabatory. In Building Machine Learning and Deep Learning Models on Google Cloud Platform (pp. 59-64). Apress, Berkeley, CA.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J.Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. and Desmaison, A., 2019. Pytorch: An imperative style, high performance deep learning library.
- J. S. Cramer, “The origins of logistic regression,” 2002.
- Al-Sarem M, Boulila W, Al-Harby M, Qadir J, Al Saeedi A (2019) Deep learning-based rumor detection on microblogging platforms: a systematic review. *IEEE Access* 7:152788–152812
- Chen W, Zhang Y, Yeo CK, Lau CT, Lee BS (2018) Unsupervised rumor detection based on users’ behaviors using neural networks. *Pattern Recogn Lett* 105:226–233. <https://doi.org/10.1016/j.patrec.2017.10.014>.
- Gargling with salt water or Vinegar 'eliminate' the COVID-19 coronavirus from the throat (n.d.) [Online]. Available: Will Gargling with Salt Water or Vinegar 'Eliminate' the COVID-19 Coronavirus? | Snopes.com