CHAPTER 1: INTRODUCTION

1. INTRODUCTION

1.1 INTRODUCTION

In today's digitally interconnected world, the proliferation of fake news and misinformation has become a pressing concern. Misleading or fabricated information can have far-reaching consequences, including the manipulation of public opinion, disruption of political processes, and damage to individual reputations. As such, the need for robust tools and techniques to detect and combat fake news is paramount. This report introduces a comprehensive analysis of one such approach, utilizing Long Short-Term Memory (LSTM) neural networks for the detection of fake news.

Fake news, often disguised as legitimate news, spreads rapidly across various online platforms. Traditional methods of fact-checking and verification struggle to keep up with the sheer volume and speed at which misinformation is disseminated. Consequently, there is an urgent need for automated and scalable solutions that can identify fake news with high accuracy.

The Long Short-Term Memory (LSTM) neural network, a subset of recurrent neural networks (RNNs), has proven to be highly effective in natural language processing tasks. LSTMs excel at modeling sequential data, making them an ideal candidate for fake news detection, as the language used in news articles and social media posts is inherently sequential and contextual.

This report will explore the application of LSTM-based models to the task of fake news detection. It will delve into the architecture and training process of LSTMs, highlighting their ability to capture the nuanced features of textual content. Additionally, we will discuss the challenges and limitations associated with this approach and explore the techniques used to mitigate them.

The report will also provide an overview of the available datasets and resources commonly used for training and evaluation, as well as the various evaluation metrics used to assess the performance of LSTM-based fake news detection models. We will discuss the importance of feature engineering, data preprocessing, and the role of embeddings in improving model accuracy.

Furthermore, we will explore the ethical considerations related to fake news detection, including privacy concerns, potential biases, and the responsibility of model developers to ensure the responsible and fair application of their technology.

In conclusion, the rising tide of fake news presents a critical challenge in the digital age, and LSTM-based models offer a promising solution to this problem. This report aims to provide a detailed exploration of the methods, challenges, and ethical considerations in utilizing LSTMs for fake news detection. By the end, readers will have a comprehensive understanding of how LSTM-based approaches can contribute to the ongoing battle against the spread of misinformation.

1.2 MOTIVATION

In this paper, the research process, technical analysis, technical linguistics work, and classifier performance and results are presented. The paper concludes with a discussion of how the current system will evolve into an influence mining system. The fake news stories that are initially seeded over social media platforms share key linguistic characteristics such as excessive use of unsubstantiated hyperbole and non-attributed quoted content. The results of a fake news identification study that documents the performance of a fake news classifier are presented and discussed in this paper.

1.3 PROBLEM DEFINITION

Fake News Detection is a critical issue in the current digital information landscape. The spread of false or misleading information, often disguised as legitimate news, can have significant societal, political, and economic implications. The problem revolves around identifying and categorizing news articles, social media posts, and online content as either genuine or fake, and various machine learning techniques, including Long Short-Term Memory (LSTM) models, have been explored as potential solutions.

The primary goal of this report is to define the problem of fake news detection and explain why LSTM-based models are relevant and promising for addressing this issue. The problem can be broken down into several key components:

- **Information Overload** The internet is flooded with information, making it increasingly challenging for individuals to discern between credible and non-credible sources. Fake news detection aims to provide an automated solution to this problem, helping users make more informed decisions about the information they consume.

- **High Stakes** Fake news can have severe consequences, including the manipulation of public opinion, the exacerbation of social divisions, and the erosion of trust in journalism. Detecting fake news is essential for safeguarding the integrity of democratic processes and maintaining public trust.

- **Rapid Dissemination** Fake news spreads quickly, aided by the viral nature of social media platforms. Traditional fact-checking methods cannot keep pace with the sheer volume and speed of misinformation. LSTM-based models are designed to process and analyze sequential data, making them suitable for real-time detection.

- **Nuanced Language** Fake news often employs sophisticated language and techniques to appear genuine. LSTM models, with their ability to capture context and sequential patterns in text, can potentially identify subtle linguistic cues that indicate the veracity of information.

- **Challenges in Data Labeling** Creating reliable labeled datasets for training fake news detection models is challenging. It requires a large, diverse dataset of both real and fake news articles, which can be labor-intensive and subjective. Ensuring the quality and representativeness of the training data is crucial.

- **Ethical Considerations** There are ethical concerns related to false positives and false negatives in fake news detection, as mislabeling legitimate content can lead to censorship and harm to freedom of speech. Striking a balance between accuracy and fairness is an ongoing challenge.

- **Generalization** The effectiveness of LSTM-based models in fake news detection depends on their ability to generalize to previously unseen or evolving types of fake news. The models should adapt to new tactics used by purveyors of fake news.

- **Deployment and Integration** Once a fake news detection model is developed, there are practical challenges in deploying and integrating it into online platforms, news aggregators, or social media networks for real-world use.

In summary, the problem of fake news detection using LSTM models is multifaceted, encompassing technical, ethical, and practical aspects. Addressing this problem involves designing, training, and deploying LSTM-based models that can effectively differentiate between credible and non-credible information sources while minimizing false positives and negatives and ensuring ethical

use in the digital landscape. This report will explore how LSTM models can contribute to this endeavor.

1.4 OBJECTIVE OF THE PROJECT

Fake news has been demonstrated to be problematic in multiple ways. It has been shown to have real influence on public perception and the ability to shape regional and national dialogue. It has harmed businesses and individuals and even resulted in death, when an individual responded to a hoax. It has caused some teenagers to reject the concept of media objectivity and many students can't reliably tell the difference between real and faked articles . It is even thought to have influenced the 2016 United States elections . Fake news can be spread deliberately by humans or indiscriminately by bot armies , with the latter giving a nefarious article significant reach. Not just articles are faked, in many cases fake, mislabeled or deceptive images are also used to maximize impact . Some contend that fake news is a "plague" on society's digital infrastructure . Many are working to combat it. Farajtabar, et al. , for example, has proposed a system based on points, while Haigh, Haigh and Kozakhave suggested the use of "peer-to-peer counter propaganda

1.5 LIMITATIONS OF THE PROJECT

While LSTM (Long Short-Term Memory) neural networks hold promise for fake news detection, it is essential to recognize their limitations. Understanding these constraints is critical for developing effective and reliable fake news detection systems. This section outlines some of the key limitations associated with using LSTMs for this purpose:

- **Limited Context** LSTMs, like other recurrent neural networks, have a finite memory of past information. They may struggle to capture long-range dependencies in text, which can be crucial for understanding the context of a news article fully.

- **Model Complexity** LSTMs are complex models, which can make them computationally expensive to train and deploy. This complexity can hinder real-time processing, especially when dealing with large volumes of text data.

- **Data Dependence** LSTM models heavily rely on the quality and quantity of training data. Inadequate or biased training data can result in models that perform poorly, especially if the data used for training does not adequately represent the diversity of fake news articles.

- **Overfitting** LSTMs can be prone to overfitting, where the model learns to memorize the training data rather than generalize from it. This can result in poor performance when applied to unseen data.
- **Lack of External Knowledge** LSTMs rely solely on the information contained within the text they are analyzing. They do not have access to external knowledge, which can be essential for verifying facts and claims made in news articles.
- **Evolving Tactics** Purveyors of fake news constantly adapt and develop new tactics to deceive detection systems. LSTM models may struggle to keep up with these evolving techniques, requiring constant updates and retraining.
- **Multimodal Content** Fake news is not limited to text; it can also include images, videos, and audio. LSTM models are primarily designed for text data and may not effectively handle these other forms of misinformation.
- **Sarcasm and Irony** LSTMs might misinterpret sarcasm, irony, or other forms of figurative language, classifying satirical content as fake news or genuine news inaccurately.
- **Low Resource Languages** LSTM models can perform well in languages with substantial training data, but they may struggle in low-resource languages with limited labeled data available for training.
- **Interpretable Outputs** LSTMs are often considered as black-box models, making it challenging to interpret why a particular decision was made. Interpreting the model's output is crucial for trust and accountability in fake news detection.
- **Ethical Concerns** Fake news detection, if not carefully designed, can have ethical implications, including the potential for censorship and bias. LSTM models may inadvertently perpetuate biases present in the training data.
- **Limited Generalization** LSTM models might not generalize well across various domains or types of content. They may perform excellently on news articles but struggle with social media posts, for instance.

In conclusion, LSTM-based fake news detection systems offer a powerful tool for addressing the issue of misinformation, but they are not without their limitations. To create more effective and ethical solutions, researchers and developers must acknowledge and address these constraints while exploring complementary approaches and techniques to enhance the accuracy, robustness, and fairness of fake news detection systems.

1.6 ORGANIZATION OF THE REPORT

This is to follow up the next chapters i.e., Chapter 2 contains the information about the system specifications. It clearly explains the libraries offered by the system. Software requirements and hardware requirements are also mentioned in the chapter. The next chapter i.e., Chapter 3 deals with the design and implementation of the project. It covers the technology that is used for the project i.e. Natural Language Tool Kit, SciKit Learn, Matplotlib. It also contains the source code of the project and the output screenshots of the project. The last chapter i.e., Chapter 4 provides the concluding information of the project. The report ends with a list of references that have been used.

CHAPTER 2: SYSTEM APPLICATIONS

2. SYSTEM APPLICATIONS

2.1 SOFTWARE SPECIFICATIONS

1. The service should be able to store the users data.
2. The data should be accessible through any devices connected to the Internet.
3. The service should be capable of synchronizing the user's data between multiple devices (notebooks, smartphones).
4. The service should preserve all historical changes(versioning).
5. Data should be shareable with other users.
6. The service should support SSO.
7. The service should be interoperable with other cloud storage services, enabling data migration from one CSP to another.

Operating System: Windows

Coding Language: Python 3.7

2.2 HARDWARE SPECIFICATIONS

Processor - Pentium–III

Speed – 2.4GHz

RAM - 512 MB(min)

Hard Disk - 20 GB

Floppy Drive - 1.44MB

Key Board - Standard Keyboard

Monitor – 15 VGAColour

CHAPTER 3 :LITERATURE SURVEY

3. LITERATURE SURVEY

3.1 INTRODUCTION

In the past, there have been researches carried out by different groups , belonging to both, industry and academia that bear similar resemblance to or are based on atopic similar to what we are doing

No.	AUTHOR (S)	TITLE OF THE PAPER	YEAR OF PUBLICATION	PROPOSED METHODOLOGY	LIMITATIONS
1	Shloka Gilda	Evaluating Machine Learning Algorithms for Fake News Detection	2017	In this paper, the author introduced the concept of the importance of NLP in stumbling across incorrect information. They have used time frequency-inverse document frequency (TF-IDF) of bigrams and probabilistic context-free grammar detection. Shloka Gilda introduced the concept of the importance of NLP in stumbling over incorrect information. They used Bi-Gram Count Vectorizer and Probabilistic Context-Free Grammar (PCFG) to detect deceptions. They examined the data set in more than one class of algorithms to find out a better model.	• Data quality, imbalances. • Generalization challenges. • Adversarial attacks. Bias and fairness concerns. • Privacy issues. • Evolving tactics. • Model interpretability. • Scalability requirements.

2	Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, Huan Liu	Fake News Detection on Social Media: A Data Mining Perspective.	2017	In this paper to detect fake news on social media, a data mining perspective is presented that includes the characterization of fake news in psychology and social theories. This article looks at two main factors responsible for the widespread acceptance of fake messages by the user which is naive realism and confirmatory bias. It proposes a general two-phase data mining framework that includes 1) feature extraction and 2) modeling, analyzing data sets, and confusion matrix for detecting fake news.	• Limited novelty and practical implementation. • Lack of real-world validation and case studies. • Insufficient details on data sources and model specifics. • Need for discussion on scalability, biases, and interpretability. • Privacy and ethical considerations should be addressed.

3	Shivam B. Parikh, Pradeep K. Atrey	Media Rich Fake News Detection: A Survey	2018	In this paper, the (multilingual) text is analyzed with the help of computational linguistics, which semantically and systematically focuses on the creation of the text. Since most publications are in the form of text, a lot of work has been done on analyzing them. Multimedia: Several forms of media are integrated into a single post. This can include audio, video, images, and graphics. This is very attractive and attracts the viewer's attention without worrying about the text. Hyperlinks allow the author of the post to refer to various sources and thus gain the trust of viewers. In practice, references are made to other social media websites, and screenshots are inserted.	• Limited emphasis on practical implementation and real-world case studies. • The survey may not comprehensively address the scalability challenges associated with media-rich fake news detection. • Ethical and privacy considerations related to handling media-rich content are not thoroughly discussed.

4	Iftikhar Ahmad, Muhammad Yousaf, Suhail Yousaf, Muhammad Ovais Ahmad	Fake News Detection using Machine Learning Ensemble Methods	2020	In this paper, The use machine learning ensemble approach for automated classification of news articles is propped. Study explores different textual properties that can be used to distinguish fake contents from real. By using those properties, A combination of different machine learning algorithms using various ensemble methods is trained and their performance on 4 real world datasets is evaluated. Experimental evaluation confirms the superior performance of the proposed ensemble learner approach in comparison to individual learners.	• Potential lack of interpretability for complex ensemble models. • The need for diverse, high-quality labeled datasets. • Challenges in dealing with evolving tactics by purveyors of fake news. • Limited discussion on model scalability and real-world deployment. • Potential for bias and fairness concerns in ensemble methods.

5	Z Khanam, B N Alwasel, H Sirafi and M Rashid	Fake News Detection Using Machine Learning Approaches	2021	This paper makes an analysis of the research related to fake news detection and explores the traditional machine learning models to choose the best, in order to create a model of a product with supervised machine learning algorithm, that can classify fake news as true or false, by using tools like python scikit-learn, NLP for textual analysis. This process will result in feature extraction and vectorization; we propose using Python scikit-learn library tper form tokenization and feature extraction of text data, because this library contains useful tools like Count Vectorizer and Tiff Vectorizer. Then, we will perform feature selection methods, to experiment and choose the best fit features to obtain the highest precision, according to confusion matrix results.	• Data quality and representativeness. • Ethical and privacy concerns. • Adversarial evasion tactics. • Scalability and real-world deployment challenges. • Limited interpretability for complex models.

Table 3.1.1 : Literature Survey

3.2 EXISTING SYSTEM

Some common existing approaches and systems

- **Fact-Checking Organizations** Fact-checking organizations, such as Snopes, PolitiFact, and FactCheck.org, manually review and verify the accuracy of news articles and claims. They publish their findings to debunk false information. These organizations play a crucial role in the fight against fake news by providing credible sources for fact-checking.

- **Machine Learning Models** Various machine learning models, including traditional classifiers and deep learning models, have been applied to fake news detection. These models analyze textual content and metadata to classify news articles as real or fake. Feature engineering, natural language processing techniques, and large labelled datasets are used to train these models.

- **Social Media Monitoring** Many fake news stories spread through social media platforms. Tools and systems have been developed to monitor social media for the rapid dissemination of misinformation. These tools can identify trending topics, monitor user engagement, and track the propagation of misleading content.

- **Source Verification Tools** These tools focus on verifying the credibility of news sources. They examine the reputation of news websites, the history of articles published by these sources, and the quality of journalism to assess the likelihood of a source producing fake news.

- **User Behavior Analysis** Some systems analyze user behaviour and interactions on social media to identify potential sources of fake news. Unusual patterns, such as bots or accounts spreading sensationalized content, can be indicators of misinformation.

- **Content-Based Analysis** Natural language processing (NLP) techniques are employed to analyze the content of news articles. Sentiment analysis, linguistic patterns, and topic modelling can reveal characteristics of fake news, such as exaggerated claims, emotional language, or sensationalism.

- **Deep Learning Models** Deep learning models, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been used for text and image analysis in fake news detection. LSTM, a type of RNN, is one of the deep learning architectures applied to sequential data for this purpose.

- **Claim-Based Fact-Checking** Some systems focus on fact-checking specific claims made in news articles or social media posts. These systems use databases of verified facts to check the accuracy of claims and statements.

- **Data Mining and Pattern Recognition** These systems employ data mining and pattern recognition techniques to identify common patterns and characteristics associated with fake news articles. This can include statistical analysis of features, such as the frequency of specific words or phrases.

- **Collaborative Approaches** Collaboration between technology companies, academic researchers, and fact-checking organizations has led to the development of shared databases and tools to combat fake news. Efforts like the "ClaimReview" schema enable fact-checkers to markup fact-checking articles for search engines and social media platforms.

It's important to note that no single system or approach is foolproof in detecting fake news. Many systems use a combination of these methods to increase accuracy and reliability. Additionally, addressing the challenges and limitations of fake news detection, such as ethical considerations and adversarial attacks, remains an ongoing area of research and development in the field.

3.3 DISADVANTAGES OF EXISTING SYSTEM

The existing systems and approaches for fake news detection have several disadvantages and limitations, which can impact their effectiveness and reliability. Some of the common disadvantages include:

- **Incompleteness of Databases** Fact-checking organizations and databases of verified facts may not cover all claims and news articles. This incompleteness limits the ability to cross-reference claims and verify the accuracy of information comprehensively.

- **Human Bias** Human fact-checkers and moderators can introduce bias, whether intentional or unintentional, into the fact-checking process. This bias can impact the objectivity and credibility of the verification process.

- **False Positives and Negatives** Machine learning models and algorithms used for fake news detection are not perfect and can produce false positives (labeling accurate information as fake) and

false negatives (failing to detect actual fake news). Striking a balance between accuracy and false alarms is challenging.

- **Adversarial Attacks** Malicious actors actively try to circumvent fake news detection systems. They can craft fake news articles specifically designed to evade detection, making it a cat-and-mouse game to keep up with evolving tactics.

- **Content Context** Fake news detection systems often focus on individual articles or claims without considering the broader context. Misleading information can be presented in a way that is difficult to identify as false when taken out of context.

- **Language and Cultural Bias** Fake news detection models developed in one language or culture may not perform well in others. Language nuances, cultural references, and context-specific language can be challenging for these systems to interpret accurately.

- **Content Format** Existing systems primarily focus on text-based content, but fake news can also be disseminated through images, videos, and audio. Detection of multimedia misinformation is more challenging and may not be as well-addressed.

- **Privacy Concerns** Analyzing social media data and user behavior for fake news detection raises privacy concerns, as user data may be collected and analyzed without their consent.

- **Scalability** The scalability of human fact-checking and manual verification processes is limited. Automation through machine learning models helps to scale the detection process, but it comes with its own set of challenges.

- **Algorithmic Fairness** Fake news detection models can inadvertently perpetuate biases present in the training data. This can result in unfair treatment or censorship of certain groups, especially if the training data is biased.

- **Content Evolution** Fake news tactics constantly evolve, and the static nature of many existing systems may struggle to adapt to new forms and strategies of misinformation.

- **Limited Generalization** Some models trained on specific datasets may not generalize well to different types of fake news or across various domains.

- **Resource Intensive** Training and deploying machine learning models, especially deep learning models, can be computationally expensive and require significant resources, making them less accessible for smaller organizations.

- **Lack of Transparency** Some fake news detection systems, especially deep learning models, are considered as black-box models, making it challenging to understand their decision-making processes and explanations for their classifications.

3.4 PROPOSED SYSTEM

In this paper author is describing concept to detect fake news from social media or document corpus using Natural Language Processing and attribution supervised learning estimator. News documents or articles will be uploaded to application and then by using Natural Language Processing to extract quotes, verbs and name entity recognition (extracting organizations or person names) from documents to compute score, verbs, quotes and name entity also called as attribution. Using supervised learning estimator we will calculate score between sum of verbs, sum of name entity and sum of quotes divided by total sentence length. If score greater than 0 then news will be consider as REAL and if less than 0 then new will be consider as FAKE.

Adavatages of Proposed System

1. It is desirable to use COX data for phylogenetic exploration.
2. We use the data of COX experimental values.
3. Security

CHAPTER 4 : DESIGN AND IMPLEMENTATION

4. DESIGN AND IMPLEMENTATION

4.1 INTRODUCTION

The design methodology describes the UML diagrams, dataset, implementation of the project, source code and output screenshots and testing. The UML diagrams include use case diagram, sequence diagram, class diagram, activity diagram, data flow component, and component diagram. The implementation of the project provides the information about details like Data Processing, Datasets, and Model (Fake News Detection). The source code is providing in Python Programming Language. The output screenshots are the outputs we get after executing the source code. The testing shows the types of testing conducted to check the performance of the project.

4.1.1 Purpose

The purpose of fake news detection using LSTM (Long Short-Term Memory) and other natural language processing (NLP) techniques is to address the growing issue of misinformation and disinformation in the digital age. Fake news detection serves several important purposes:

1. Preserve Information Integrity Fake news can spread rapidly and harm society by disseminating false or misleading information. Fake news detection helps maintain the integrity of information sources, ensuring that news is accurate and reliable.

2. Protect Public Trust Trust in the media and news sources is essential for a well-informed and functioning society. Detecting and addressing fake news helps protect the public's trust in journalism and authoritative sources.

3. Combat Disinformation Fake news is often used to manipulate public opinion, influence elections, and create confusion. By detecting and mitigating fake news, we can reduce the impact of disinformation campaigns.

4. Promote Fact-Checking Fake news detection encourages fact-checking and responsible journalism. News outlets are incentivized to verify information before publishing, leading to more reliable reporting.

5. Minimize Harm Some fake news stories can have real-world consequences, such as causing panic, harm to reputations, or economic losses. Detecting fake news can minimize these harms.

6. Improve Media Literacy Fake news detection can be used as an educational tool to increase media literacy. Teaching individuals how to critically evaluate news sources and content can empower them to make informed decisions.

7. Support Social Media Platforms Social media platforms and news aggregators can use fak news detection to reduce the spread of false information on their platforms. This helps maintain the quality and credibility of content shared online.

8. Legal and Ethical Considerations In some regions, spreading false information may have legal consequences. Fake news detection can support legal actions against those who deliberately disseminate misinformation.

9. Research and Analysis Detecting fake news is an active area of research in NLP and machine learning. It contributes to the development of more advanced AI models for text classification and understanding.

10. Business Applications Organizations can use fake news detection to protect their brands and reputation. They can monitor news and social media for false information that may affect them.

4.1.2 Overall Description

1. **Data Collection** Gather a dataset of news articles, comprising both real and fake news examples.

2. Data Preprocessing

- Tokenize the text to break it into words or subwords.
- Pad or truncate sequences to a fixed length for model input.
- Encode labels (real or fake) into numerical values.

3. Word Embeddings Convert words into dense vector representations using pre-trained word embeddings (e.g., Word2Vec, GloVe) or train your own embeddings.

4. LSTM Model

- Build a Long Short-Term Memory (LSTM) neural network architecture.

- Use word embeddings as input.
- Stack one or more LSTM layers for capturing sequential patterns.

5. Training

- Split the dataset into training and validation sets.
- Train the LSTM model on the training data, using binary cross-entropy loss.
- Monitor and adjust hyperparameters like learning rate and batch size.

6. Evaluation

- Assess the model's performance using evaluation metrics like accuracy, precision, recall, and F1-score.
- Utilize a validation set to prevent overfitting.

7. Testing Apply the trained model to new, unseen news articles for fake news detection.

8. Post-processing Set a threshold for classifying articles as real or fake based on model output probabilities.

9. Model Fine-tuning Adjust model architecture, hyperparameters, or collect more data if the performance is unsatisfactory.

10. Deployment Integrate the model into an application or platform for real-time fake news detection.

4.2 UML DIAGRAMS

I. USE CASE DIAGRAMS

A use case diagram in the Unified Modeling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. The roles of the actors in the system can be depicted.

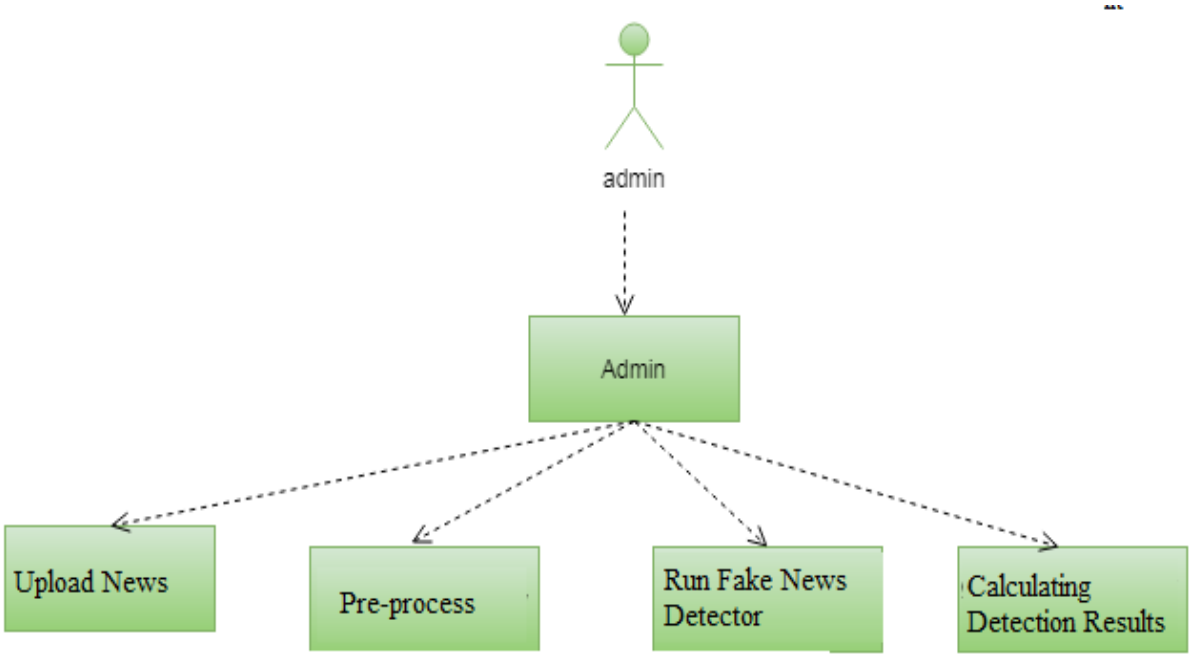


Fig 4.2.1 : Use Case Diagram

II. SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

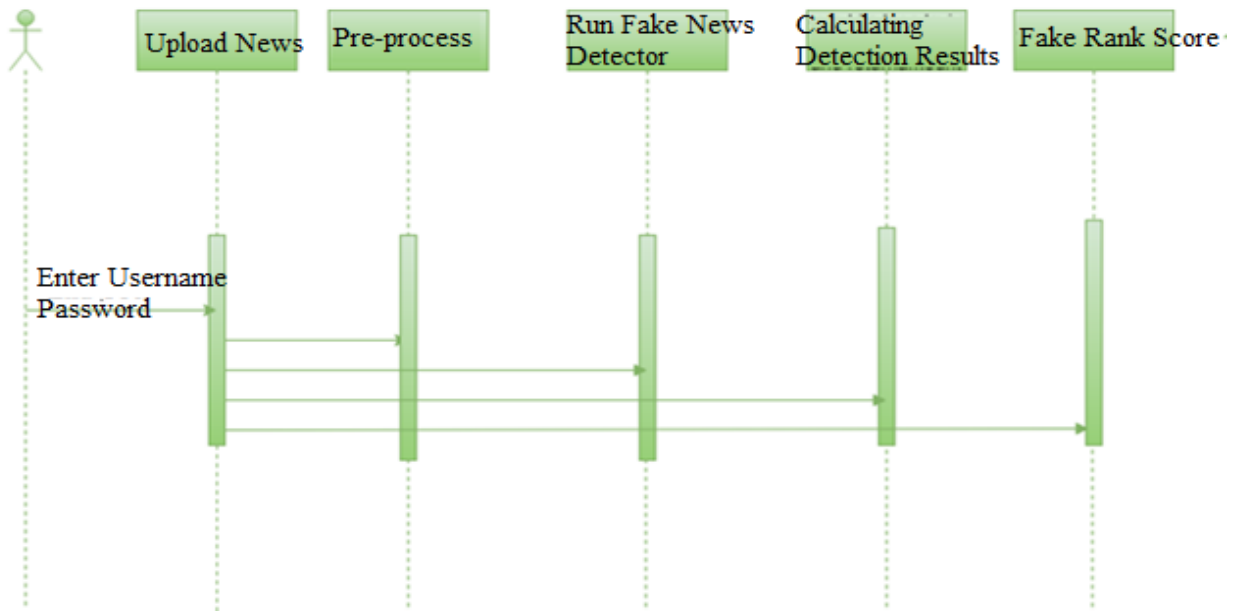


Fig 4.2.2 : Sequence Diagram

III. CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

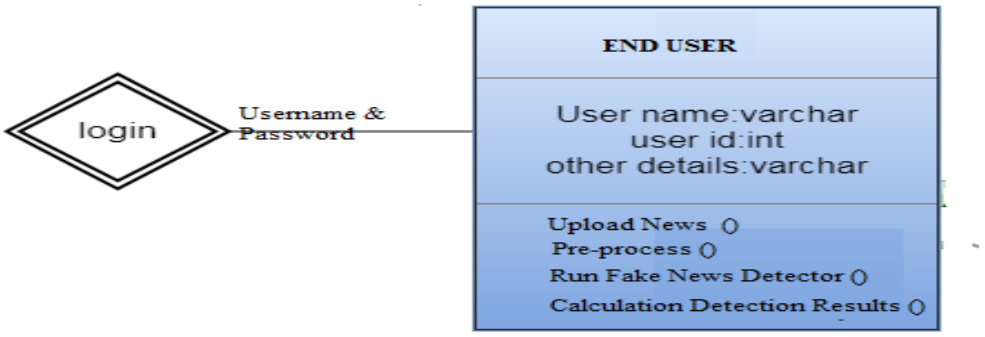


Fig 4.2.3 : Class Diagram

IV. ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

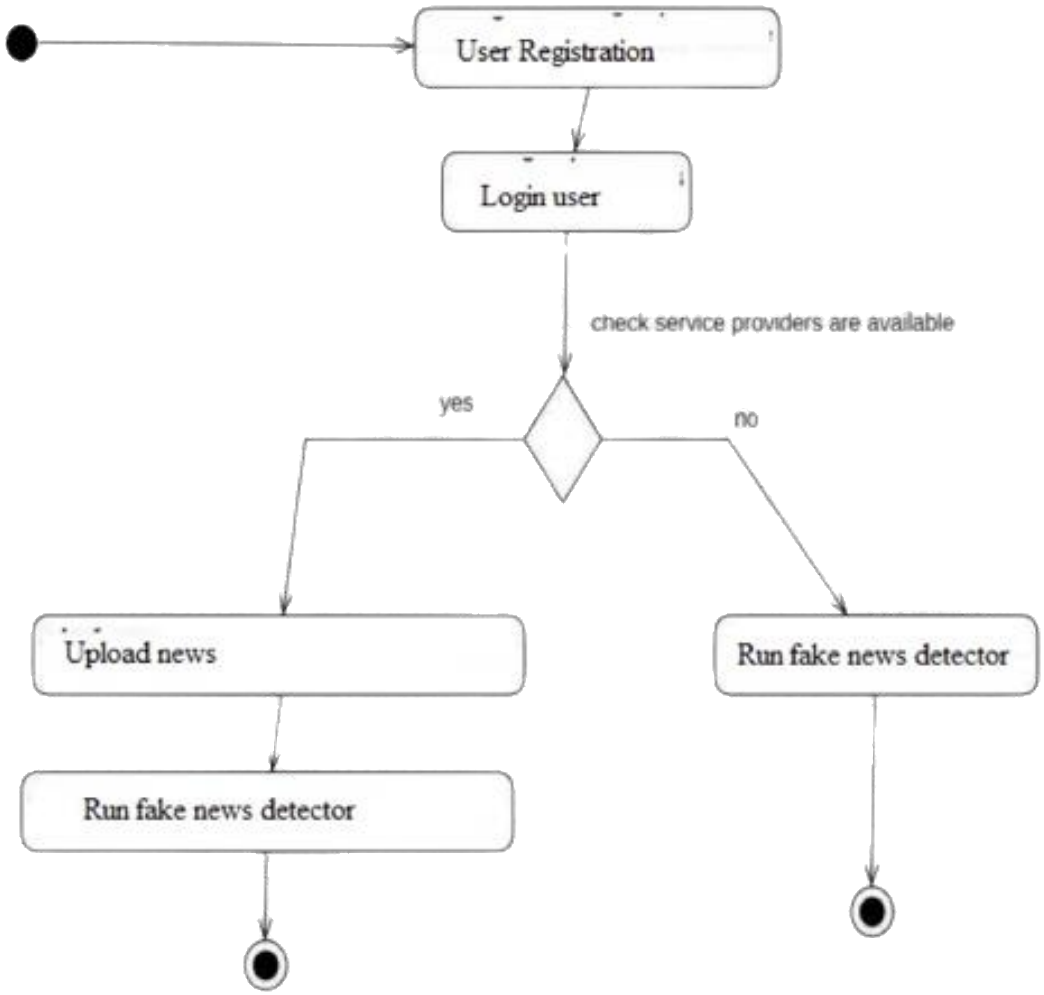


Fig 4.2.4 : Activity Diagram

V. DATA FLOW DIAGAM

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.. DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between User and System designer. Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams. DFD has often been used due to the following reasons:

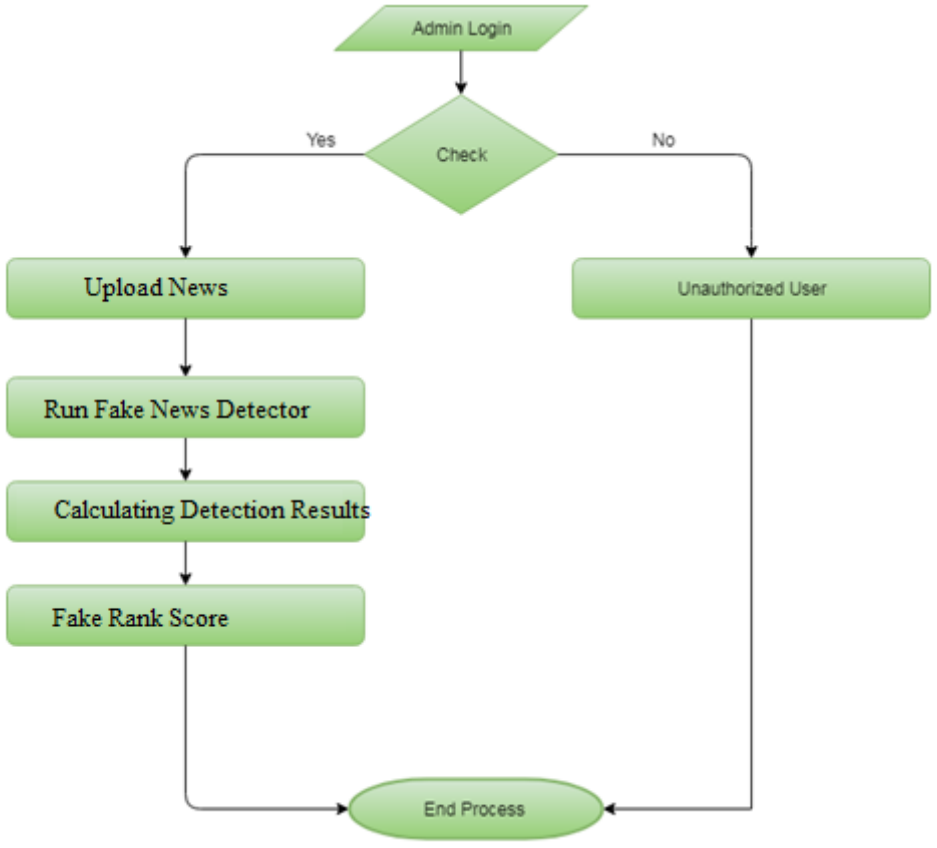


Fig 4.2.5 : Data Flow Diagram

VI. COMPONENT DIAGRAM

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

UML Component diagrams are used in modelling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

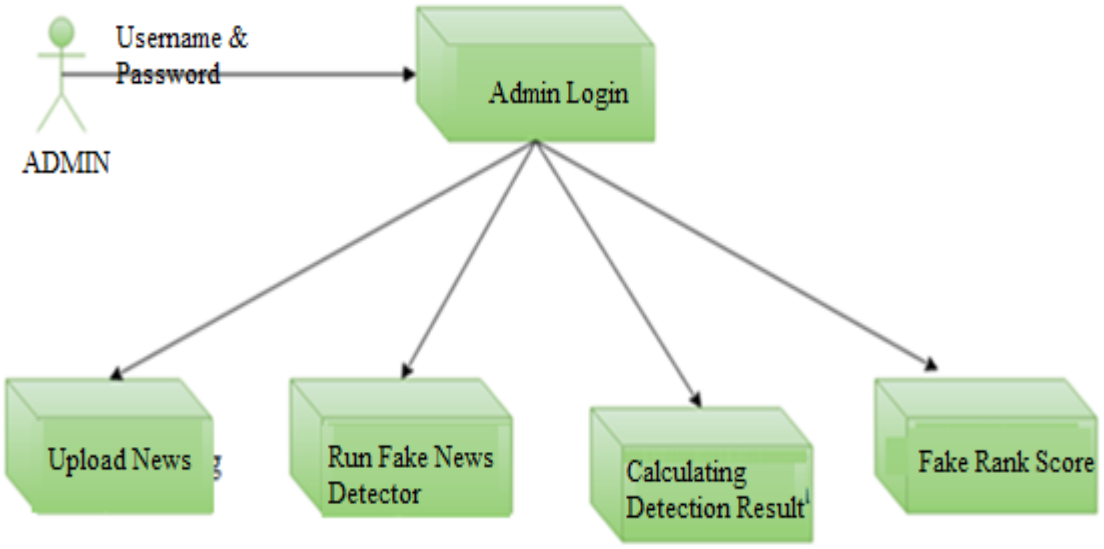


Fig 4.2.6 : Compoent Diagram

4.3 MODULE DESCRIPTION

I. Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

II. Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

III. Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics.

IV. Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits.Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

V. Scikit-Learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

VI. NLTK

NLTK (Natural Language Toolkit) is a popular Python library used for working with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. NLTK is widely used in natural language processing (NLP) and computational linguistics for various text analysis and language understanding tasks.

Key Features of NLTK

- **Tokenization** NLTK provides functions for splitting text into words or sentences, which is essential for many NLP tasks.
- **Stopwords** It includes a list of common stopwords (e.g., "the," "and," "in") and tools for filtering them out in text analysis.

- **Stemming and Lemmatization** NLTK offers tools for reducing words to their root forms, which can be useful for text normalization.
- **Part-of-Speech Tagging** You can use NLTK to tag words in a text with their parts of speech (e.g., noun, verb, adjective).
- **Parsing** NLTK includes parsers for syntax analysis, including chart parsers and recursive descent parsers.
- **Named Entity Recognition (NER)** It offers tools for identifying and classifying named entities (e.g., persons, organizations, locations) in text.

NumPy: Base n-dimensional array package

SciPy: Fundamental library for scientific computing

Matplotlib: Comprehensive 2D/3D plotting

IPython: Enhanced interactive console

Sympy: Symbolic mathematics

Pandas: Data structures and analysis

Extensions or modules for SciPy care are conventionally named SciKits.

4.4 DATASET

The Dataset is divided into two files, one file is used for training and the other is used for testing.

The training dataset is named“news.csv” and contains the following attributes

- **id**: unique ID for a news article
- **title**: the title of a news article
- **author**: author of the news article
- **text**: the text of the article; could be incomplete

- **label**: a label that marks the article as potentially unreliable
  - 1: unreliable
  - 0: reliable

The testing training dataset is named **"testnews.csv"** with all the same attributes as train.csv without the label. The input of news articles is given through a text file which is named **"text"** which you can upload to test the data.

4.5 IMPLEMENTATION

Implementing a fake news detection system using LSTM (Long Short-Term Memory) and NLP techniques involves several steps, including data preprocessing, model building, training, and evaluation.

Below is a step-by-step guide on how to implement fake news detection using LSTM:

Step 1: Data Preparation

I. **Data Collection** Gather a dataset of news articles labeled as real and fake news. You can find publicly available datasets or create your own.

II. **Data Preprocessing**:

- Clean the text data by removing special characters, punctuation, and stop words.
- Tokenize the text into words.
- Pad or truncate sequences to a fixed length if needed.
- Convert text data to numerical sequences (word embeddings).

Step 2: Data Splitting

I. **Split the Data** into training and testing datasets, 80% of the data for training and 20% for testing.

Step 3: Model Building

I. **Import Libraries**

Import necessary libraries, including TensorFlow or Keras.

II. **Create an LSTM Model**

Build an LSTM-based model for text classification.

```
from keras.models import Sequential

from keras.layers import Embedding, LSTM, Dense

model = Sequential()

model.add(Embedding(input_dim, output_dim, input_length=max_sequence_length))

model.add(LSTM(100))

model.add(Dense(1, activation='sigmoid'))
```

Adjust the model's hyperparameters based on the specific dataset and requirements.

Step 4: Model Compilation and Training

I. **Compile the Model**

Specify the loss function (binary cross-entropy for binary classification) and the optimizer (e.g., Adam). Choose appropriate metrics for evaluation.

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

II. **Train the Model**

Fit the model to your training data. Specify the batch size and the number of training epochs.

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), batch_size=batch_size, epochs=num_epochs)
```

Step 5: Model Evaluation

I. Evaluate the Model on the testing data to calculate accuracy, loss, and other relevant metrics.

loss, accuracy = model.evaluate(X_test, y_test)

print(f'Accuracy: {accuracy * 100:.2f}%')

Step 6: Inference

After training, model is used to make predictions on new text data to classify whether it's real or fake news.

Step 7: Fine-Tuning and Optimization

Different model architectures, hyperparameters, and pre-trained word embeddings (e.g., GloVe or Word2Vec) are used to train to improve the model's performance.

Step 8: Deployment

Set up a web application or API that allows users to input text and receive predictions.

4.6 SOURCE CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
import matplotlib.pyplot as plt
import numpy as np
from tkinter import ttk
from tkinter import filedialog
import pandas as pd
from sklearn.model_selection import train_test_split
from string import punctuation
```

```
from nltk.corpus import stopwords
import nltk
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers.core import Dense,Activation,Dropout
from sklearn.preprocessing import OneHotEncoder
import keras.layers
from keras.models import model_from_json
import pickle
import os
from sklearn.preprocessing import normalize

from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, LSTM

main = Tk()
main.title("DETECTION OF FAKE NEWS THROUGH IMPLEMENTATION OF DATA
SCIENCE APPLICATION")
main.geometry("1300x1200")

global filename
global X, Y
global tfidf_X_train, tfidf_X_test, tfidf_y_train, tfidf_y_test
global tfidf_vectorizer
global accuracy,error
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
textdata = []
labels = []
global classifier
```

```
def cleanPost(doc):
    tokens = doc.split()
    table = str.maketrans('', '', punctuation)
    tokens = [w.translate(table) for w in tokens]
    tokens = [word for word in tokens if word.isalpha()]
    tokens = [w for w in tokens if not w in stop_words]
    tokens = [word for word in tokens if len(word) > 1]
    tokens = [lemmatizer.lemmatize(token) for token in tokens]
    tokens = ' '.join(tokens)
    return tokens

def uploadDataset():
    global filename
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="TwitterNewsData")
    textdata.clear()
    labels.clear()
    dataset = pd.read_csv(filename)
    dataset = dataset.fillna(' ')
    for i in range(len(dataset)):
        msg = dataset.get_value(i, 'text')
        label = dataset.get_value(i, 'target')
        msg = str(msg)
        msg = msg.strip().lower()
        labels.append(int(label))
        clean = cleanPost(msg)
        textdata.append(clean)
        text.insert(END,clean+" ==== "+str(label)+"\n")

def preprocess():
    text.delete('1.0', END)
    global X, Y
```

```
    global tfidf_vectorizer
    global tfidf_X_train, tfidf_X_test, tfidf_y_train, tfidf_y_test
    stopwords=stopwords = nltk.corpus.stopwords.words("english")
    tfidf_vectorizer = TfidfVectorizer(stop_words=stopwords, use_idf=True,
ngram_range=(1,2),smooth_idf=False, norm=None, decode_error='replace', max_features=200)
    tfidf = tfidf_vectorizer.fit_transform(textdata).toarray()
    df = pd.DataFrame(tfidf, columns=tfidf_vectorizer.get_feature_names())
    text.insert(END,str(df))
    print(df.shape)
    df = df.values
    X = df[:, 0:df.shape[1]]
    X = normalize(X)
    Y = np.asarray(labels)
    le = LabelEncoder()
    Y = le.fit_transform(Y)
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    Y = Y.reshape(-1, 1)
    print(X.shape)
    encoder = OneHotEncoder(sparse=False)
    #Y = encoder.fit_transform(Y)
    X = X.reshape((X.shape[0], X.shape[1], 1))
    print(Y)
    print(Y.shape)
    print(X.shape)
    tfidf_X_train, tfidf_X_test, tfidf_y_train, tfidf_y_test = train_test_split(X, Y, test_size=0.2)
    text.insert(END,"\n\nTotal News found in dataset : "+str(len(X))+"\n")
    text.insert(END,"Total records used to train machine learning algorithms :
"+str(len(tfidf_X_train))+"\n")
```

```
    text.insert(END,"Total records used to test machine learning algorithms  :
"+str(len(tfidf_X_test))+"\n")


def runLSTM():
    text.delete('1.0', END)
    global classifier
    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            classifier = model_from_json(loaded_model_json)
        classifier.load_weights("model/model_weights.h5")
        classifier._make_predict_function()
        print(classifier.summary())
        f = open('model/history.pckl', 'rb')
        data = pickle.load(f)
        f.close()
        acc = data['accuracy']
        acc = acc[9] * 100
        text.insert(END,"LSTM Fake News Detection Accuracy : "+str(acc)+"\n\n")
        text.insert(END,'LSTM Model Summary can be seen in black console for layer details\n')
        with open('model/model.txt', 'rb') as file:
            classifier = pickle.load(file)
        file.close()
    else:
        lstm_model = Sequential()
        lstm_model.add(LSTM(128, input_shape=(X.shape[1:]), activation='relu',
return_sequences=True))
        lstm_model.add(Dropout(0.2))

        lstm_model.add(LSTM(128, activation='relu'))
        lstm_model.add(Dropout(0.2))
```

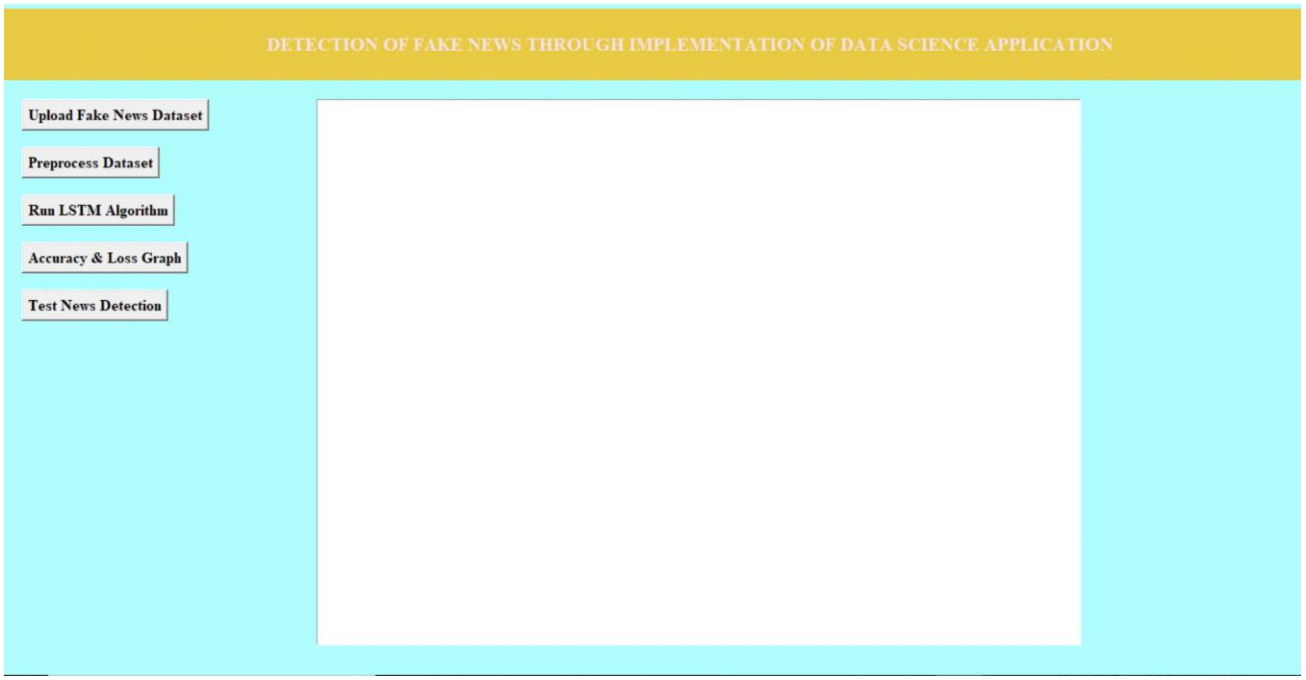
```
        lstm_model.add(Dense(32, activation='relu'))
        lstm_model.add(Dropout(0.2))
        lstm_model.add(Dense(2, activation='softmax'))
        lstm_model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
        hist = lstm_model.fit(X, Y, epochs=10, validation_data=(tfidf_X_test, tfidf_y_test))
        classifier = lstm_model
        classifier.save_weights('model/model_weights.h5')
        model_json = classifier.to_json()
        with open("model/model.json", "w") as json_file:
            json_file.write(model_json)
        accuracy = hist.history
        f = open('model/history.pckl', 'wb')
        pickle.dump(accuracy, f)
        f.close()
        acc = accuracy['accuracy']
        acc = acc[9] * 100
        text.insert(END,"LSTM Accuracy : "+str(acc)+"\n\n")
        text.insert(END,'LSTM Model Summary can be seen in black console for layer details\n')
        print(lstm_model.summary())


def graph():
    f = open('model/history.pckl', 'rb')
    data = pickle.load(f)
    f.close()
    acc = data['accuracy']
    loss = data['loss']
    plt.figure(figsize=(10,6))
    plt.grid(True)
    plt.xlabel('Epcchs')
    plt.ylabel('Accuracy/Loss')
```

```
    plt.plot(acc, 'ro-', color = 'green')
    plt.plot(loss, 'ro-', color = 'blue')
    plt.legend(['Accuracy','Loss'], loc='upper left')
    #plt.xticks(wordloss.index)
    plt.title('LSTM Model Accuracy & Loss Graph')
    plt.show()

def predict():
    testfile = filedialog.askopenfilename(initialdir="TwitterNewsData")
    testData = pd.read_csv(testfile)
    text.delete('1.0', END)
    testData = testData.values
    testData = testData[:,0]
    print(testData)
    for i in range(len(testData)):
        msg = testData[i]
        msg1 = testData[i]
        print(msg)
        review = msg.lower()
        review = review.strip().lower()
        review = cleanPost(review)
        testReview = tfidf_vectorizer.transform([review]).toarray()
        predict = classifier.predict(testReview)
        print(predict)
        if predict == 0:
            text.insert(END,msg1+" === Given news predicted as GENUINE\n\n")
        else:
            text.insert(END,msg1+" == Given news predicted as FAKE\n\n")

font = ('times', 15, 'bold')
title = Label(main, text='DETECTION OF FAKE NEWS THROUGH IMPLEMENTATION OF
DATA SCIENCE APPLICATION')
```

```
title.config(bg='gold2', fg='thistle1')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 13, 'bold')
ff = ('times', 12, 'bold')

uploadButton = Button(main, text="Upload Fake News Dataset", command=uploadDataset)
uploadButton.place(x=20,y=100)
uploadButton.config(font=ff)

processButton = Button(main, text="Preprocess Dataset", command=preprocess)
processButton.place(x=20,y=150)
processButton.config(font=ff)

dtButton = Button(main, text="Run LSTM Algorithm", command=runLSTM)
dtButton.place(x=20,y=200)
dtButton.config(font=ff)

graphButton = Button(main, text="Accuracy & Loss Graph", command=graph)
graphButton.place(x=20,y=250)
graphButton.config(font=ff)

predictButton = Button(main, text="Test News Detection", command=predict)
predictButton.place(x=20,y=300)
predictButton.config(font=ff)

font1 = ('times', 12, 'bold')
text=Text(main,height=30,width=100)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
```

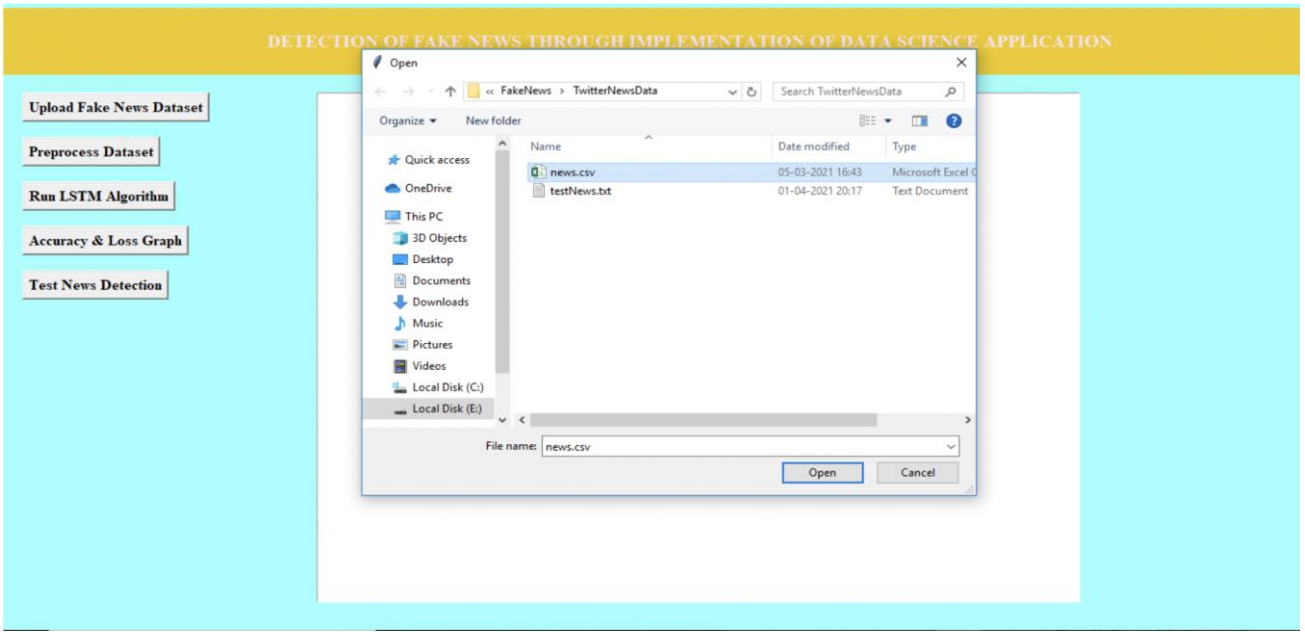
```
text.place(x=330,y=100)
text.config(font=font1)

main.config(bg='DarkSlateGray1')
main.mainloop()
```

4.7 OUTPUT SCREENSHOT



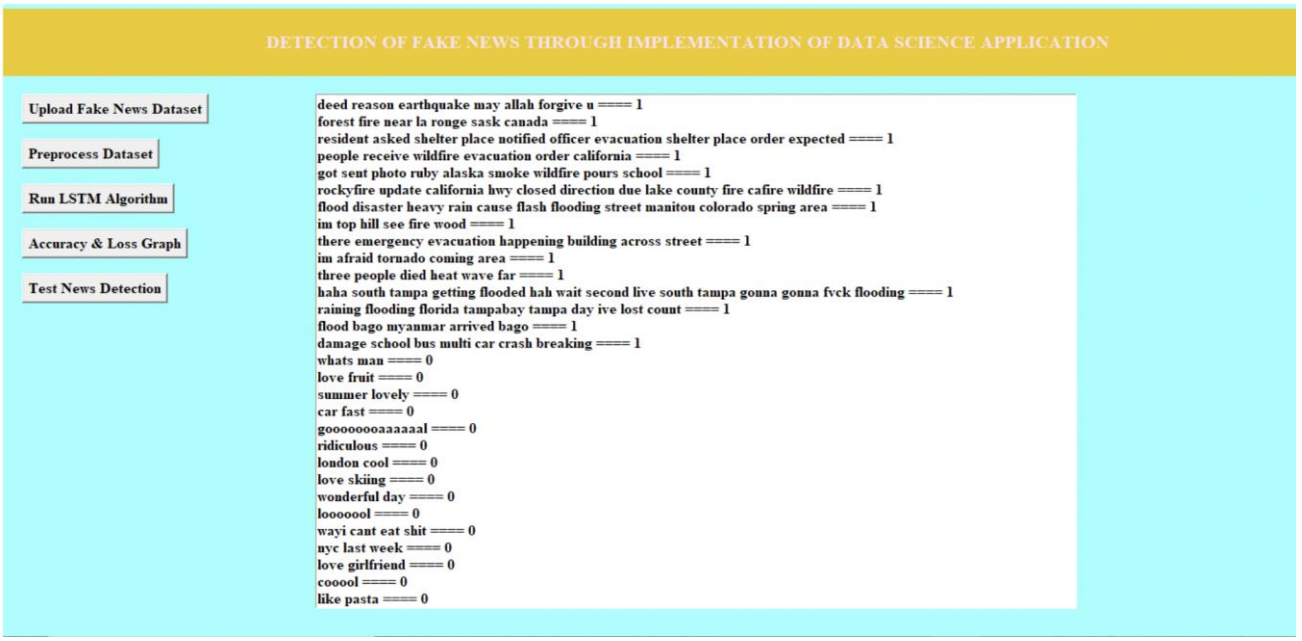
To run the project double-click on ‘run.bat ’file to get the below screen.

Fig 4.7.1 : User Interface



In the above screen click on the 'Upload Fake News Dataset 'button to upload the dataset.

Fig 4.7.2 : Uploading Training Dataset



In the above screen select and upload the 'news.csv 'file and then click on the 'Open 'button to load the dataset and to get the below screen

Fig 4.7.3 : Converting Strings to Numeric Data

In the above screen dataset loaded and then in the text area we can see all news text with the class label as 0 or 1 now click on the 'Preprocess Dataset & Apply NGram 'button to convert the above string data to a numeric vector and to get the below screen.

In the below screen, all news words are put in the column header and if that word appears in any row then that row's column will be changed with the word count and if not appear then 0 will be put in the column. In above screen shows some records from total of 7612 news records and in the bottom lines we can see the dataset contains a total 7613 records and then application uses 80% (6090 news records) for training and then using 20% (1523 news records) for testing and now dataset is ready with numeric record and now click on 'Run LSTM Algorithm 'button to train above dataset with LSTM and then build LSTM model and then calculate accuracy and error rate.

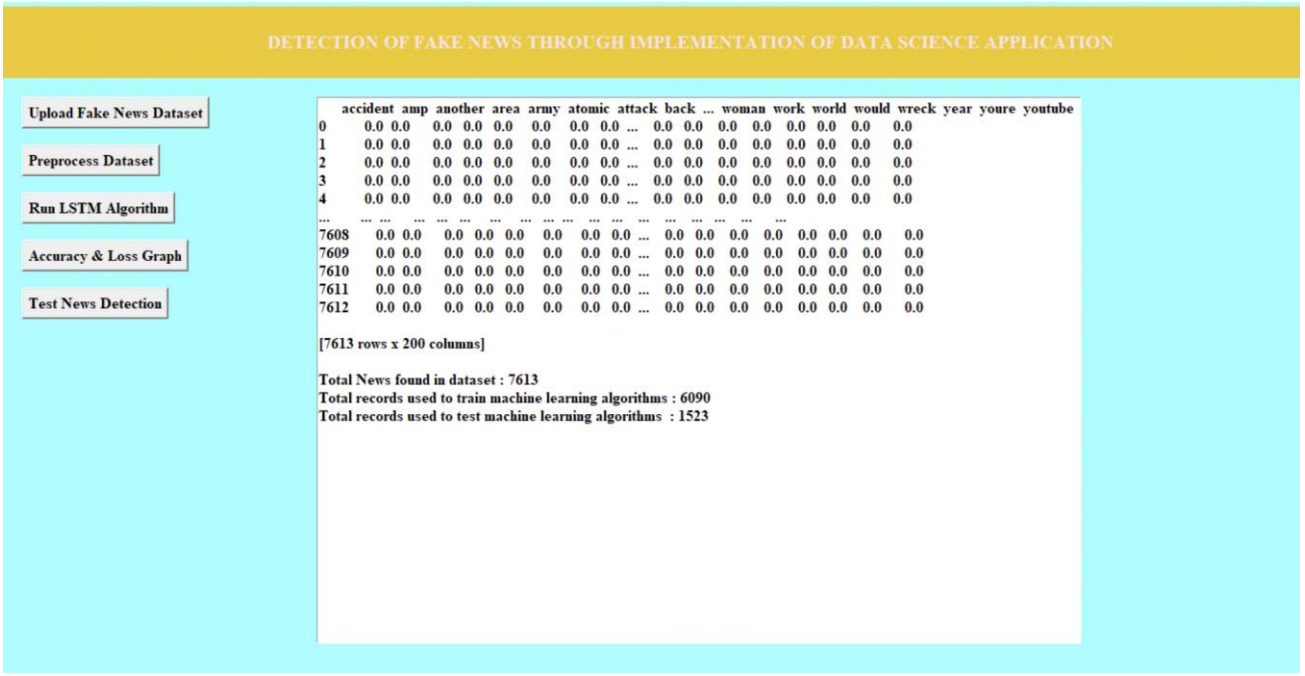


Fig 4.7.4 : Calculating Word Count

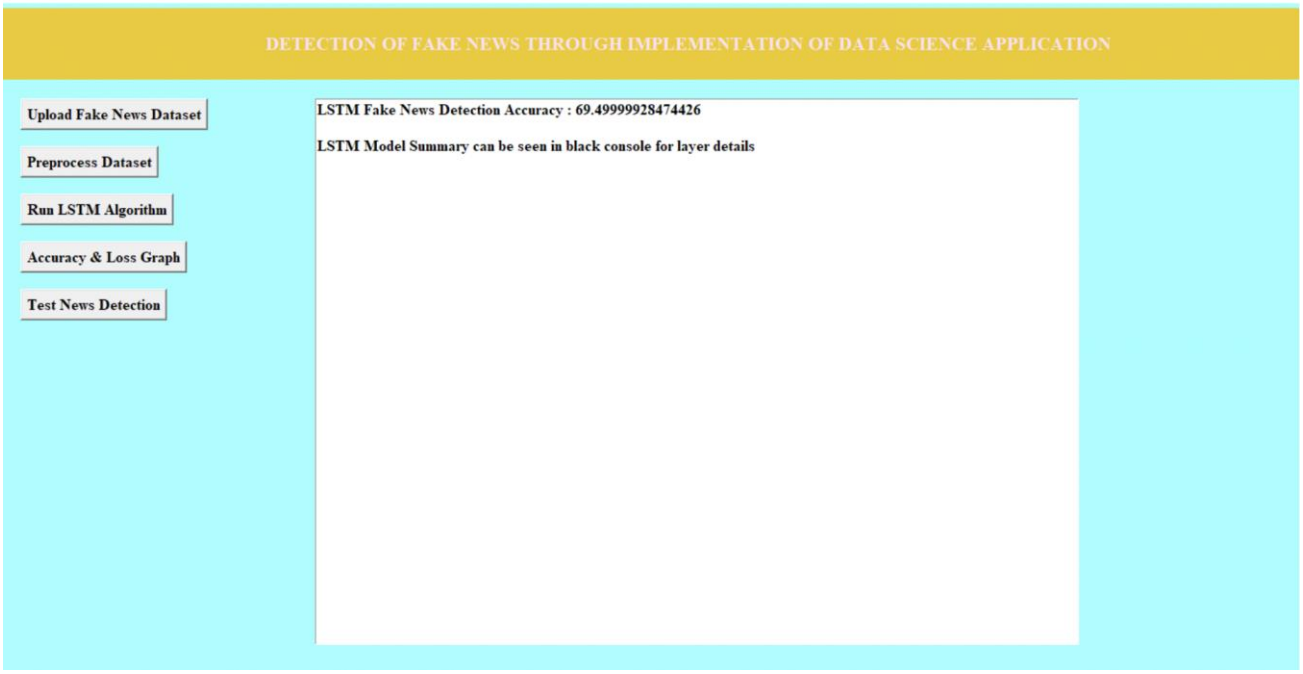


Fig 4.7.5 : Accuracy Prediction

In above screen LSTM model is generated and we got its prediction accuracy as 69.49% and we can see below console to see LSTM layer details.

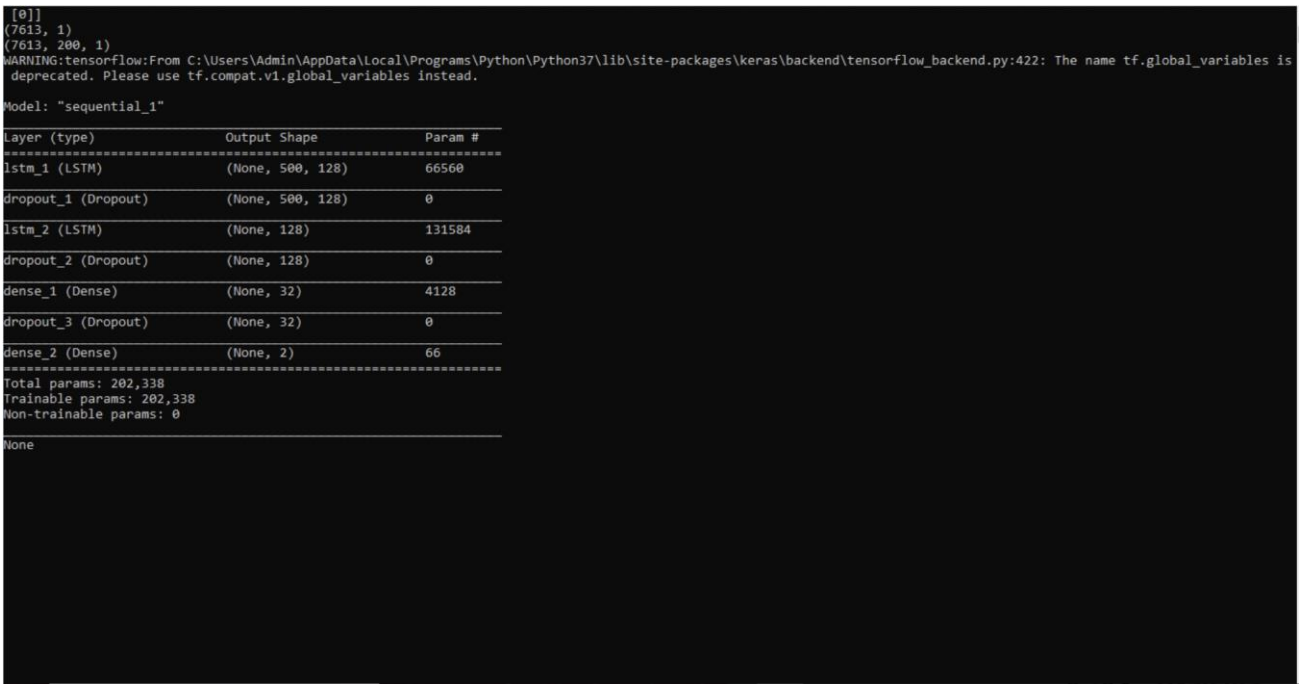


Fig 4.7.6 : LSTM Layer Creation

In above screen different LSTM layers are created to filter input data to get efficient features for prediction. Now click on ‘Accuracy & Loss Graph ’button to get LSTM graph.

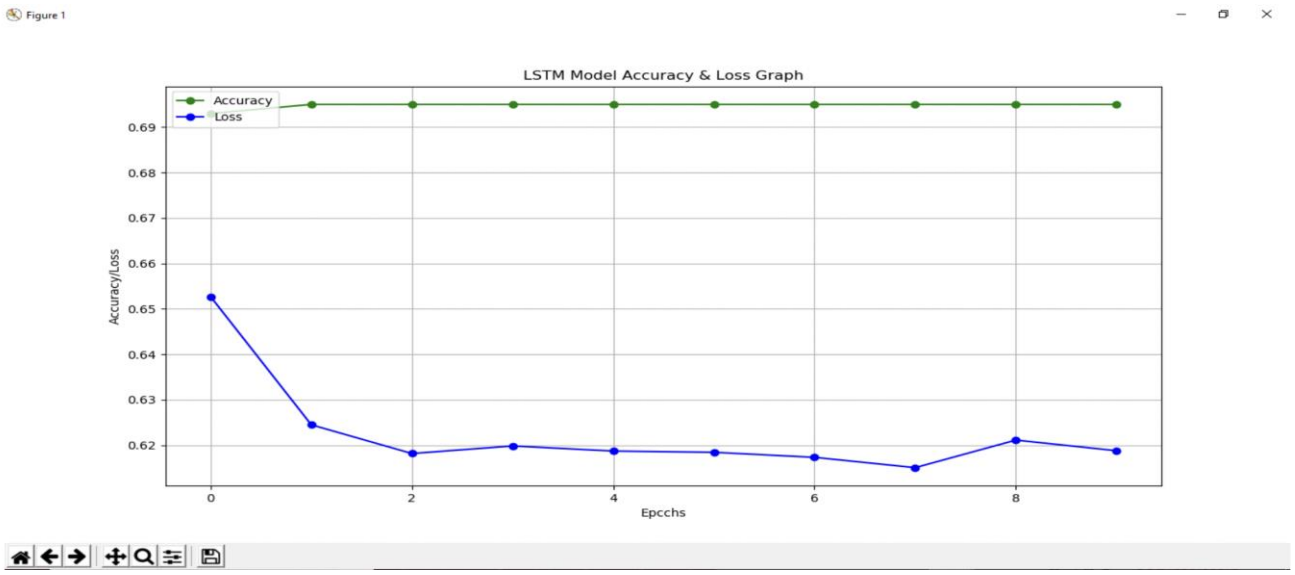
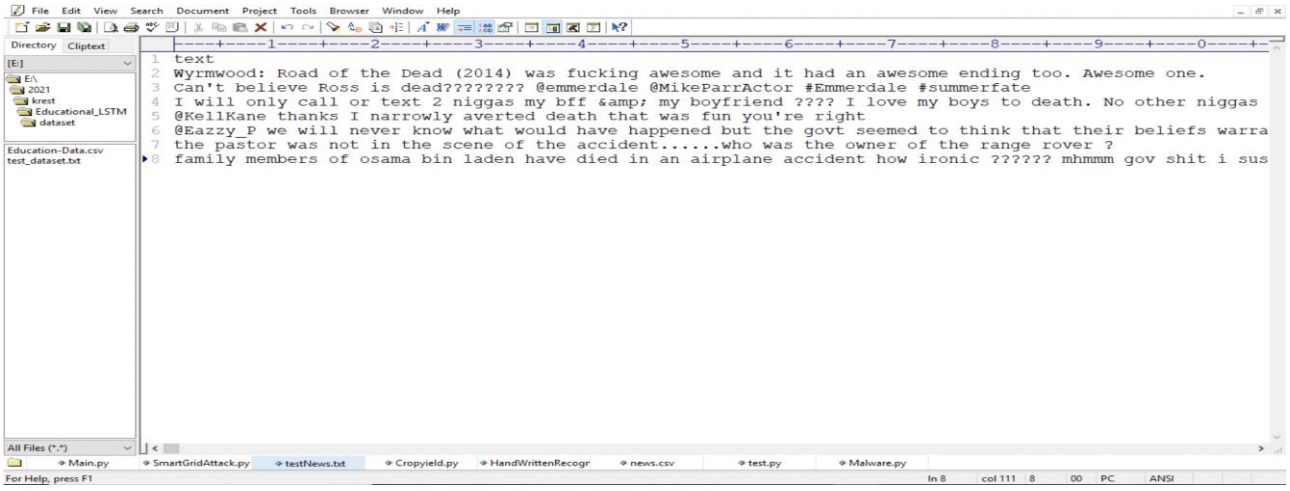


Fig 4.7.7 : LSTM Model Accuracy & Loss Graph

In above graph x-axis represents epoch/iterations and y-axis represents accuracy and loss value and green line represents accuracy and blue line represents loss value and at each increasing epoch loss values get decrease and accuracy reached to 70%. Now click on ‘Test News Detection ’button to upload some test news sentences and then application predict whether that news is genuine or fake. In below



test news dataset we can see only TEXT data no class label and LSTM will predict class label for that test news.

Fig 4.7.8 : Input Data

In above screen in test news we have only one column which contains only news 'TEXT 'and after applying above test news we will get prediction result.

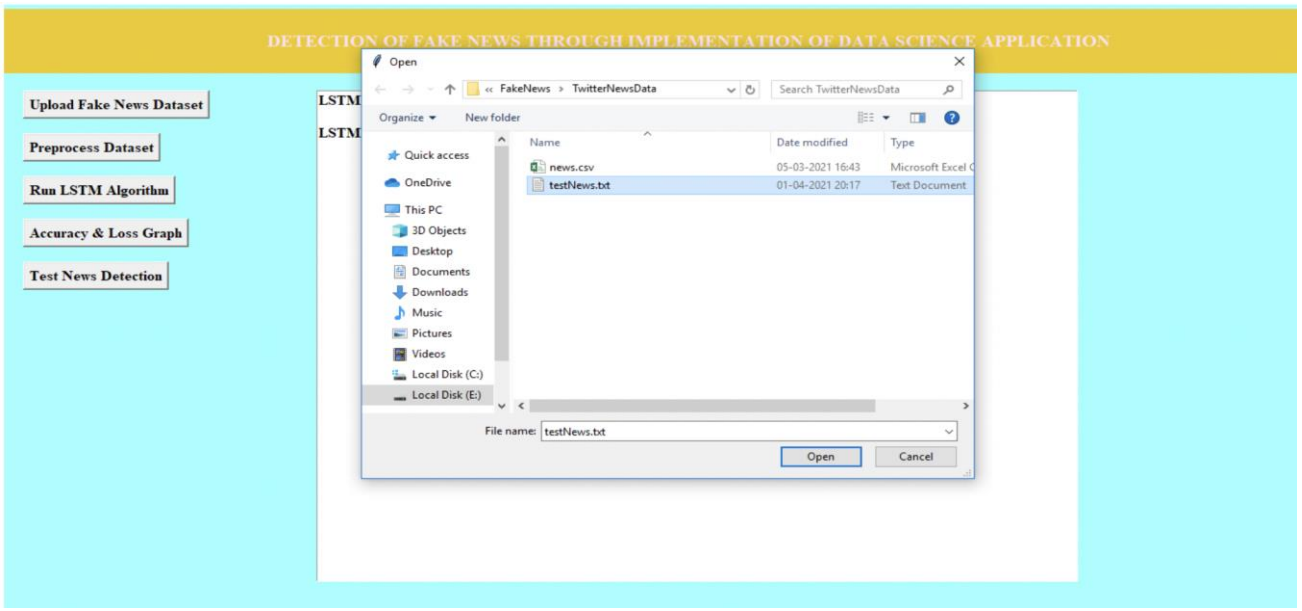


Fig 4.7.9 : Uploading Testing Data

In above screen selecting and uploading 'testNews.txt 'file and then click on 'Open 'button to load data and to get below prediction result.

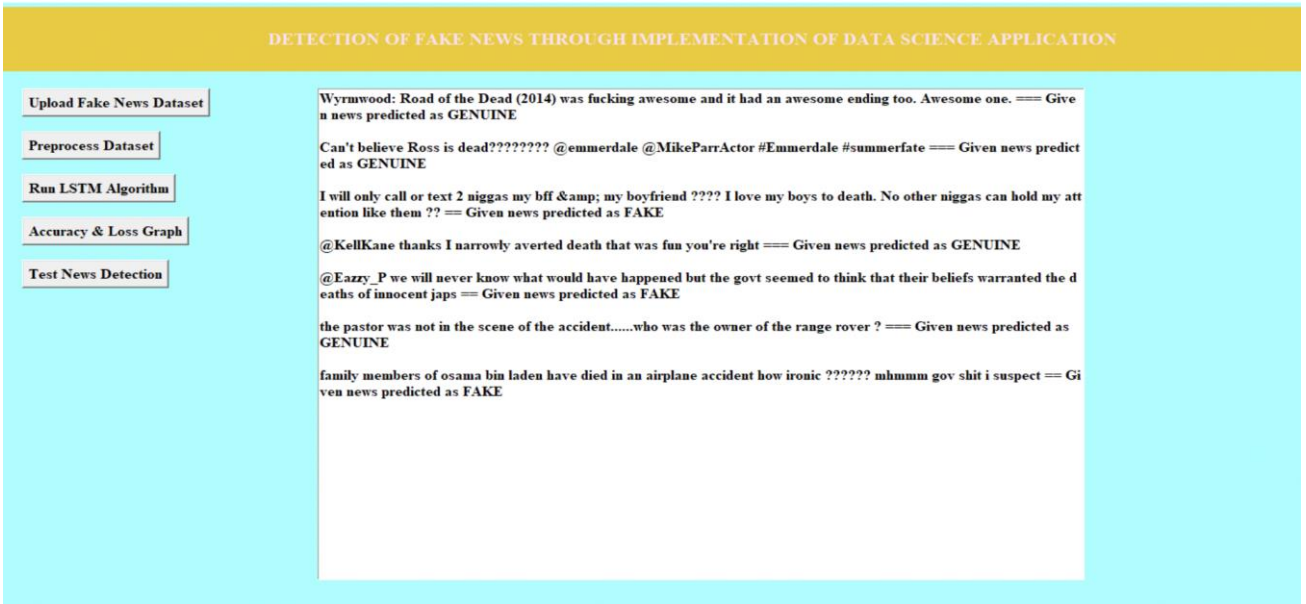


Fig 4.7.10 : News Prediction Output

In above screen before dashed symbols we have news text and after dashed symbol application predict news as 'FAKE or GENUINE'. After building model when we gave any news text then LSTM will check whether more words belongs to genuine or fake category and whatever category get more matching percentage then application will predict that class label.

4.8 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the sftware system meets its requirements and user expectations and does not fail unacceptably. There are various types of tests. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at the component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfied, as shown by successful unit testing, the combination of components is correct and consistent.

Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input identified classes of valid input must be accepted.

Invalid Input identified classes of invalid input must be rejected.

Functions identified functions must be exercised.

Output identified classes of application outputs must be exercised.

Systems/Procedures interfacing systems or procedures must be invoked.

Organization and preparation of functional tests are focused on requirements, key functions, or special test cases. In addition, systematic coverage about identifying Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester knows the inner workings, structure and language of the software or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, like most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated as, as black box. you cannot—see‖intoit. The test provides inputs and responds to outputs without considering how the software works.

I. UnitTesting

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test Strategy and Approach

Field testing will be performed manually and functional tests will be written in detail.

Test Objectives

- All field entries must workproperly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be Tested

Verify that the entries are of the correct format

No duplicate entries should be allowed

All links should take the user to the correct page.

II. Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects were encountered.

III. Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects were encountered.

CHAPTER 5 : CONCLUSION

5. CONCLUSION

CONCLUSION

This paper presented the results of a study that produced a limited fake news detection system. The work presented herein is novel in this topic domain in that it demonstrates the results of a full-spectrum research project that started with qualitative observations and resulted in a working quantitative model. The work presented in this paper is also promising, because it demonstrates a relatively effective level of machine learning classification for large fake news documents with only one extraction feature. Finally, additional research and work to identify and build additional fake news classification grammars is ongoing and should yield a more refined classification scheme for both fake news and direct quotes.

FUTURE ENHANCEMENT

The work presented in this paper is also promising, because it demonstrates a relatively effective level of machine learning classification for large fake news documents with only one extraction feature. Finally, additional research and work to identify and build additional fake news classification grammars is ongoing and should yield a more refined classification scheme for both fake news and direct quotes.

REFERENCES

REFERENCES

[1] Fake News Detection using Machine Learning Ensemble Methods - Iftikhar Ahmad, Muhammad Yousaf, Suhail Yousaf, Muhammad Ovais Ahmad, Complexity, vol. 2020, Article ID 8885861, 11 pages, 2020.
https://www.researchgate.net/publication/346262009_Fake_News_Detection_Using_Machine_Learning_Ensemble_Methods

[2] Evaluating Machine Learning Algorithms for Fake News Detection - by Shlok Gilda in the 2017 IEEE 15th Student Conference on Research and Development (SCOReD), December 2017, pp.110-115
https://ieeexplore.ieee.org/document/8305411

[3] Fake News Detection on Social Media: A Data Mining Perspective –Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang and Huan Liu. ACM SIGKDD Explorations Newsletter Volume 19 Issue 1 June 2017 pp 22–36
https://www.researchgate.net/publication/318981549_Fake_News_Detection_on_Social_Media_A_Data_Mining_Perspective

[4] Media Rich Fake News Detection: A Survey –Shivam B. Parikh and Pradeep K. Atrey. published in 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR) DOI: 10.1109/MIPR.2018.00093
https://ieeexplore.ieee.org/document/8397049/authors#authors

[5] Fake News Detection Using Machine Learning Approaches - Z Khanam, B N Alwasel, H Sirafi and M Rashid Published under license by IOP Publishing Ltd Citation Z Khanam et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1099 012040 DOI 10.1088/1757-899X/1099/1/012040
https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012040