CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

In navigating the intricate landscape of fake news detection, this chapter delves into the methodology employed for the research titled "Fake News Detection and Profiling System with Word Embeddings and CNN." Aiming to contribute to the evolving field of misinformation mitigation, the chapter outlines the strategic approach adopted to design and implement a robust system capable of effectively discerning fake news within the vast expanse of digital information.

In the pursuit of accuracy and nuance, the research strategically centers on the integration of Word Embeddings and Convolutional Neural Networks (CNNs). These foundational elements are meticulously selected to form the backbone of the proposed solution. Word Embeddings bring semantic richness to the analysis, allowing the system to decipher intricate linguistic nuances inherent in news articles. Simultaneously, the inclusion of CNNs addresses the spatial intricacies of textual data, offering a means to extract and interpret patterns that may elude traditional methodologies.

Emphasizing the importance of a data-driven approach, the chapter unfolds the methodology, starting with the acquisition of a diverse dataset. The chosen dataset, comprising both real and fake news sources, is carefully curated to represent the dynamic nature of misinformation circulating online. Subsequently, the pre-processing and cleaning of textual data become paramount, laying the groundwork for meaningful analysis and attribute extraction.

3.2 Research Process

The flowchart outlines a process for data pre-processing, feature extraction and engineering, model training, and evaluation in machine learning. It starts with data collection from the web or search engines and involves labelling the collected data. The data then undergoes pre-processing and is divided into training and testing datasets. Features are extracted and selected, with an option for dimensionality reduction. The training data is resampled if necessary before being used to train a classification model. The model's performance is evaluated based on precision, recall, F1-score, etc.

The image depicts a flowchart outlining a machine learning process. It begins with "Data Collection" from Web/Search Engines leading to an imbalanced dataset. "Labelling" follows Data Collection as part of the Data Pre-processing stage. "Feature Extraction and Engineering" involves Data Preparation leading to Feature Selection. There's an optional "Dimensionality Reduction" step involving Numeric (Integer, Float) or Categorical (Nominal/Ordinal/Boolean) types of data. Pearson's or Chi-Squared tests are used for Feature Selection. The original imbalanced dataset is separated into 5 folds: 4 folds for Training Data and 1 fold for Testing Data as part of "5 fold cross validation". "Re-sampling" (sampling/under-sampling/random over-sampling) can be applied to the Training Data if needed. Classification models like Logistic Regression, Naive Bayes, Random Forests, SVM are used in the "Classification" step. Model Performance is evaluated using metrics like Precision & Accuracy before it becomes a Trained Model ready to respond to User's Query.

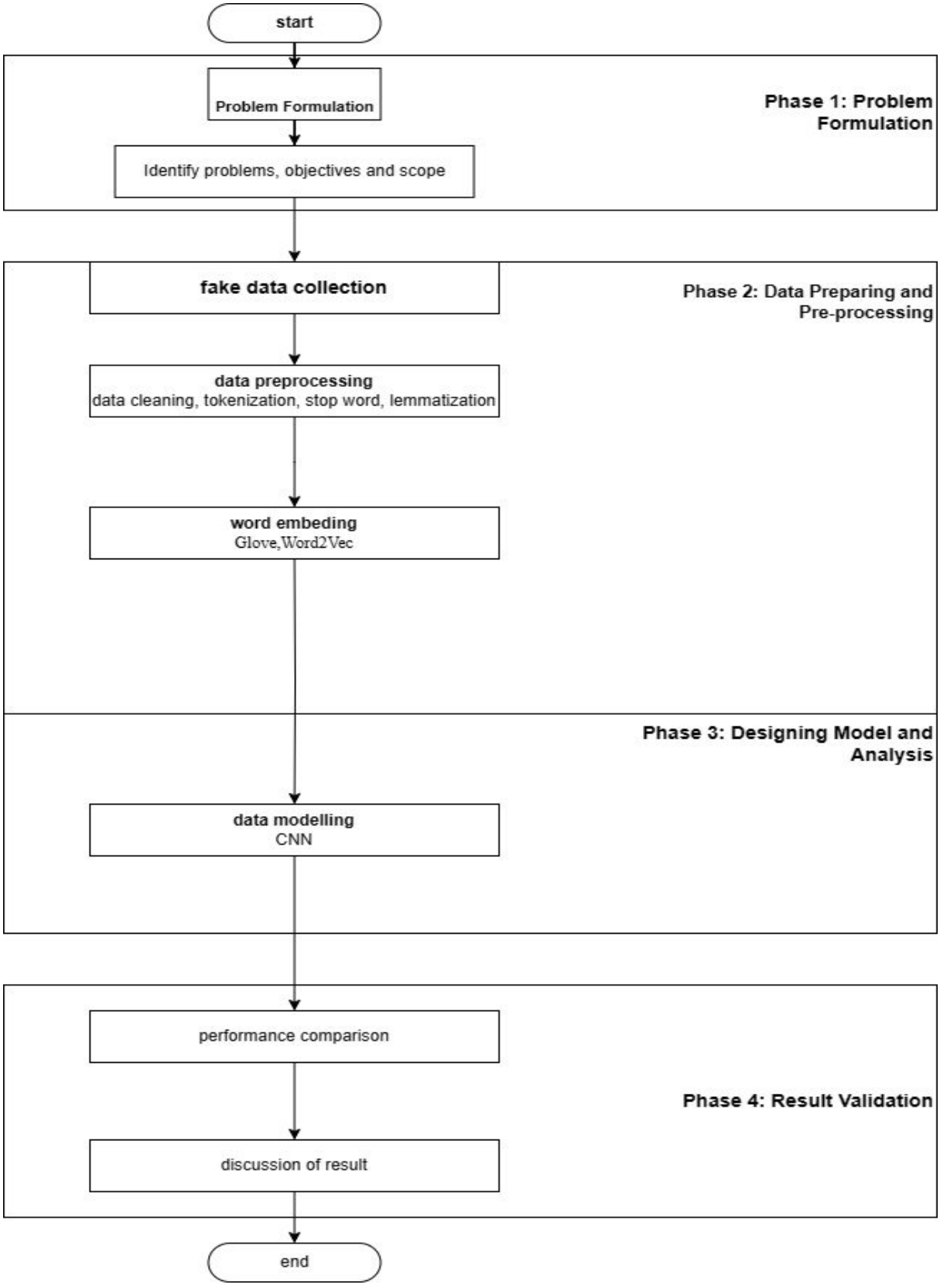

Figure 3.1 Project framework

3.2.1 Phase 1: Problem Formulation

Detecting fake news within the vast digital landscape is a complex challenge, requiring the development of a robust system capable of identifying linguistic patterns indicative of misinformation. This involves leveraging machine learning techniques to discern subtle cues and contextual nuances while adapting to evolving tactics employed by purveyors of fake news.

Profiling fake news articles adds another layer to the challenge, necessitating the extraction of relevant attributes to gain insights into classes, locations, timing, and involved individuals. Classifying fake news into distinct categories, implementing Named Entity Recognition (NER), and exploring techniques to profile temporal and spatial aspects form integral components of this endeavour.

The project aims to integrate advanced techniques by proposing a solution that combines Word Embedding and Convolutional Neural Networks (CNNs). Word Embedding capture semantic richness, enhancing linguistic context understanding, while CNNs extract spatial features and patterns crucial for identifying subtle cues indicative of fake news.

Addressing ethical considerations is paramount, involving the implementation of guidelines to ensure user privacy, data protection, and responsible information dissemination. A user-friendly interface is crucial, necessitating the design of an intuitive platform for users to input news articles, explore insights, and engage with the system, accommodating varying levels of technical expertise. Ensuring generalization and scalability is pivotal for real-world applicability, requiring the model's adaptability to new patterns and scalability for handling large volumes of data in real-time. Mitigating adversarial threats adds a layer of complexity, demanding exploration of techniques to identify and counteract adversarial attacks, safeguarding the system's robustness. The ensuing sections of the project will delve into the methodology, implementation, and evaluation of the proposed solution in response to these identified challenges.

3.2.2 Phase 2: Data Preparing and Pre-processing

Ensuring that the raw dataset is transformed into a clean, structured, and analytically usable form. This section details the systematic approach taken to handle the dataset for the "Fake News Detection and Profiling System with Word Embedding and CNN. The process is started with the data collection from the kaggle. Followed with the cleaning the text data. Then data pre-processing such as tokenization, stop word removal and lemmatization."

3.2.2.1 Data Collection

The research begins with the acquisition of a diverse dataset encompassing both real and fake news articles. The dataset is carefully selected to reflect the multifaceted nature of misinformation circulating in online platforms. The dataset is obtained from public data provider which is kaggle (https://www.kaggle.com/competitions/nlp-getting-started). The dataset is about the tweet that related to disaster. The dataset consists of 7613 tweets data with the label 1 as disaster tweet and 0 as non-disaster tweet with id, keyword, and location for each tweet.

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1
...	...	...	...	...	...
7608	10869	NaN	NaN	Two giant cranes holding a bridge collapse int...	1
7609	10870	NaN	NaN	@aria_ahrary @TheTawniest The out of control w...	1
7610	10871	NaN	NaN	M1.94 [01:04 UTC]?5km S of Volcano Hawaii: htt...	1
7611	10872	NaN	NaN	Police investigating after an e-bike collided ...	1
7612	10873	NaN	NaN	The Latest: More Homes Razed by Northern Calif...	1

7613 rows × 5 columns

Figure 3.2

3.2.2.2 Data cleaning

The initial task after loading the data is checking the quality of data. Missing value is one indicator of data quality. In this dataset there are 2533 missing value on the location and 61 missing value on keyword. But because in this project is only focus on the text data, the missing value can be ignored.

```
df.isnull().sum()
```

```
id             0
keyword       61
location    2533
text           0
target         0
dtype: int64
```

Figure 3. 3 Checking missing values

The raw data consist of messy text related data including the mention tag (@), the html tag for link and any symbol. The cleaning process is performed by defining a a function that include the data frame and the text as an input. Then the cleaning process will be using regex method. With re.sub means that changing the first input with an empty characters.

```
# cleaning data from url, hastag, symbol
def clean_text(df,text):
    df['text_clean'] = df[text].apply(lambda x: re.sub(r"http\S+", '',x))
    df['text_clean'] = df['text_clean'].apply(lambda x: re.sub(r"http", '',x))
    df['text_clean'] = df['text_clean'].apply(lambda x: re.sub(r"@\S+", '',x))
    df['text_clean'] = df['text_clean'].apply(lambda x: re.sub(r"[^A-Za-z0-9(),!?@\'\`\"\_\n]", ' ',x))
    df['text_clean'] = df['text_clean'].apply(lambda x: re.sub(r"@", "at",x))
    df['text_clean'] = df['text_clean'].str.replace('#','')
    df['text_clean'] = df['text_clean'].str.lower()
```

Figure 3.4 Data cleaning

The cleaned data then will be added into a new columns called text_clean as shown in the figure 3.5.

	id	keyword	location	text	target	text_clean
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1	our deeds are the reason of this earthquake m...
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1	forest fire near la ronge sask canada
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1	all residents asked to 'shelter in place' are ...
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	13,000 people receive wildfires evacuation or...
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1	just got sent this photo from ruby alaska as ...
...	...	...	...	...	...	...
7608	10869	NaN	NaN	Two giant cranes holding a bridge collapse int...	1	two giant cranes holding a bridge collapse int...
7609	10870	NaN	NaN	@aria_ahrary @TheTawniest The out of control w...	1	the out of control wild fires in california ...
7610	10871	NaN	NaN	M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt...	1	m1 94 01 04 utc ?5km s of volcano hawaii
7611	10872	NaN	NaN	Police investigating after an e-bike collided ...	1	police investigating after an e bike collided ...
7612	10873	NaN	NaN	The Latest: More Homes Razed by Northern Calif...	1	the latest more homes razed by northern calif...

7613 rows × 6 columns

Figure 4.5 Cleaned data

3.2.2.3 Tokenization

Text tokenization basically is splitting the sentence into the tokens or words. The library from Natural Language Tool Kit or NLTK is used, And the tokenize.word_tokenize is used for tokenizing the sentence.

```
from nltk.tokenize import word_tokenize
```

```
df['text_clean'] = df['text_clean'].apply(lambda x: word_tokenize(x))
```

Figure 3.6 Word tokenization process

3.2.2.4 Stop Word Removal

In the Natural Language Processing task, stop words such as a, I, the in, of, etc. sometimes doesn't giving valuable information to the model. Thus, in this project the stop words is removed. The stop word library is collected from nltk. The stop word removel is performed using nltk.corpus.stopwords with the English language as the input of the function.

```
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\iqbal\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
df['text_clean'] = df['text_clean'].apply(lambda x: [i for i in x if i not in stopwords.words('english')])
```

Figure 3.7 Stop word removal process

The result of removing stop word can be seen in the figure 4.7. for instance, the data number 4 that said 'Just got sent…' the word 'just' is omitted because that word is a stop word.

	id	keyword	location	text	target	text_clean
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1	[deeds, reason, earthquake, may, allah, forgiv...
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1	[forest, fire, near, la, ronge, sask, canada]
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1	[residents, asked, 'shelter, place, ', notifie...
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	[13,000, people, receive, wildfires, evacuatio...
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1	[got, sent, photo, ruby, alaska, smoke, wildfi...

Figure 3.8 Stop word removal result

3.2.2.5 Lemmatization

Lemmatization is the process of converting a words into a lemma. In another words, it change the words into the dictionary form. The lemmatization performed using nltk library specially Word Lemmatizer. The Word Lemmatizer take a part of speech and the word as an input. Thus the part of speech is obtained using pos_tag function.

```
lemmatizer = WordNetLemmatizer()
```

```
# create a function to extract part of speech (POS)
def get_pos(word):
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {'V':wordnet.VERB,
                'N':wordnet.NOUN,
                'J':wordnet.ADJ,
                'R':wordnet.ADV}
    return tag_dict.get(tag,wordnet.NOUN)
```

```
# Lemmatization
df['text_clean'] = df['text_clean'].apply(lambda x:[lemmatizer.lemmatize(i, get_pos(i)) for i in x])
```

Figure 3.9 Lemmatization process

The result of the lemmatization is stored in text_clean columns. The example of the lemmatization shown in the data on the row two, with the word asked converted into ask.

	id	keyword	location	text	target	text_clean
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1	[deed, reason, earthquake, may, allah, forgive...
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1	[forest, fire, near, la, ronge, sask, canada]
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1	[resident, ask, 'shelter, place, ', notify, of...
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	[13,000, people, receive, wildfire, evacuation...
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1	[get, sent, photo, ruby, alaska, smoke, wildfi...
...	...	...	...	...	...	...
7608	10869	NaN	NaN	Two giant cranes holding a bridge collapse int...	1	[two, giant, crane, hold, bridge, collapse, ne...
7609	10870	NaN	NaN	@aria_ahrary @TheTawniest The out of control w...	1	[control, wild, fire, california, even, northe...
7610	10871	NaN	NaN	M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt...	1	[m1, 94, 01, 04, coordinated universal time, ?...
7611	10872	NaN	NaN	Police investigating after an e-bike collided ...	1	[police, investigate, e, bike, collide, car, i...
7612	10873	NaN	NaN	The Latest: More Homes Razed by Northern Calif...	1	[late, home, raze, northern, california, wildf...

7613 rows × 6 columns

Figure 3.10

3.2.2.6 Word embedding

After the text is cleaned and converted into its lemma form, then the text data need to convert into numerical representation. The traditional methods such as bag of words and TF-IDF treat each words like totally different entity. While in the reality, some words have a correlation with each other. For instance, king is always man, queen is always women. In words embedding, it can measure the similarity between words. Word2Vec and GloVe are some of most popular word embedding methods that will be describe in this section.

3.2.2.6.1 Word2Vec

The fundamental idea behind Word2Vec is to represent words as vectors in such a way that the geometric relationships between these vectors reflect the semantic relationships between the corresponding words. The key intuition is that words with similar meanings should have similar vector representations, and their vectors should be close together in the vector space. There are two main architectures for training Word2Vec models:

1. Continuous Bag of Words (CBOW): In this architecture, the model is trained to predict a target word based on its context. The context is defined by the surrounding words in a given window. The input to the model is a set of context words, and the output is the probability distribution over the vocabulary for the target word.

2. Skip-gram: In this architecture, the model is trained to predict the context words (surrounding words) given a target word. The input is a target word, and the model aims to predict the context words within a certain window around the target word.

3.2.2.6.2 GloVe

GloVe, which stands for Global Vectors for Word Representation, is an unsupervised learning algorithm for obtaining vector representations (embeddings) of words. Developed by researchers at Stanford University, GloVe aims to capture the semantic relationships between words by representing them as vectors in a continuous vector space.

Here's a simplified overview of how GloVe works:

1. Word Co-occurrence Matrix: GloVe starts by constructing a word co-occurrence matrix. This matrix represents the frequency of word co-occurrences in a given context window. Rows and columns correspond to words, and each cell (i, j) in the matrix represents how often word i appears in the context of word j.

2. Objective Function: The core idea behind GloVe is to learn word representations such that their dot product equals the logarithm of the word's probability of co-occurrence. The objective function is designed to minimize the difference between the dot product of the word vectors and the logarithm of the observed word co-occurrence probabilities.

3. Training: The model is trained using gradient descent to minimize the objective function. During training, the word vectors are adjusted iteratively to improve the fit between the observed and predicted co-occurrence probabilities.

4. Word Embeddings: Once the training is complete, the learned word vectors represent the semantic relationships between words. These vectors encode information about how words are related based on their co-occurrence patterns in the training data.

5. Similarity and Analogies: The resulting word embeddings can be used to measure semantic similarity between words. Words with similar meanings will have similar vector representations, and vector operations can be performed to capture analogies (e.g., "king" - "man" + "woman" ≈ "queen").

3.2.3 Phase 3: Designing Model and Analysis

The heart of the research lies in the design and development of machine learning models, particularly focusing on Word Embedding and Convolutional Neural Networks (CNNs). This section delineates the systematic approach undertaken to construct robust models for the "Fake News Detection and Profiling System." The research leverages Word embedding to imbue the models with semantic understanding of the textual content. Techniques such as Word2Vec or GloVe are employed to represent words in a continuous vector space, capturing intricate relationships between them. This process transforms the textual data into numerical vectors, facilitating the models in discerning subtle semantic nuances. The core architecture of the Convolutional Neural Network (CNN) is designed to capture spatial features within the textual data. The research configures the CNN to employ convolutional layers that systematically scan and identify patterns within the Word embedding. Max-pooling layers are strategically placed to extract the most salient features, creating a hierarchical representation of the input data.

The research prioritizes interpretability and explain ability in model outputs. Techniques such as attention mechanisms within the CNN are employed to highlight regions

of importance in the input data, enhancing the transparency of the model's decision-making process. The design and analysis phase marks a pivotal juncture in the research, where sophisticated models are crafted and fine-tuned to detect and profile fake news articles. Through a combination of advanced techniques, including Word embedding and CNNs, the research aims to push the boundaries of fake news detection, offering not only accurate classification but also nuanced attribute extraction for a more profound understanding of misinformation.

By rigorously testing the developed system on the provided Kaggle dataset, the project aims to validate the effectiveness of the Word embedding and CNNs model in identifying fake news articles. The outcomes of this testing phase contribute valuable information for refining the model and ensuring its reliability in real-world scenarios.

3.2.4 Phase 4: Result Validation

At this stage, focus will be on the testing accuracy rate and loss value of the classifier will be calculated. During the learning process, the testing dataset serves as a validation set to obtain an unbiased assessment of correctness.

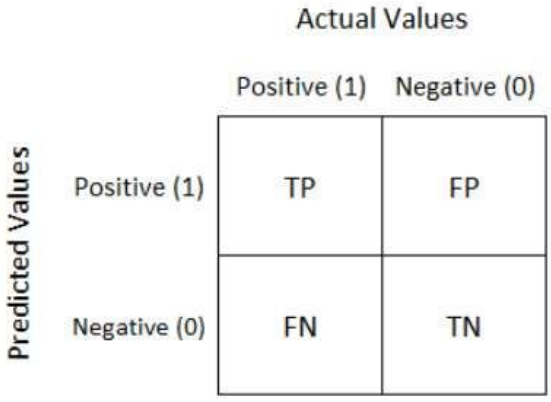

Figure 3.11 Confusion matrix

The training of the designed models involves the utilization of labeled datasets encompassing both real and fake news articles. The training process tunes the model

parameters to optimize its ability to discern patterns indicative of misinformation. Techniques like backpropagation and optimization algorithms are employed to iteratively adjust the model weights. To assess the efficacy of the designed models, a comprehensive set of evaluation metrics is employed. Metrics such as accuracy, precision, recall, and F1 score are calculated to gauge the model's performance in correctly classifying real and fake news articles. Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) are utilized for a nuanced understanding of model performance. Beyond binary classification, the research emphasizes attribute extraction from news articles. The models are trained to extract attributes such as the class of fake news (e.g., financial, war-related), location, timing, and relationships between articles. Attribute analysis provides valuable insights into the nuanced characteristics of misinformation, contributing to a comprehensive profiling system.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

$$precission = \frac{TP}{TP + FP} \tag{3.2}$$

$$recall = \frac{TP}{TP + FN} \tag{3.3}$$

$$f1\ score = \frac{2}{\frac{1}{precission} + \frac{1}{recall}} \tag{3.4}$$

3.3 Summary

This chapter outlines the systematic approach undertaken to realize the objectives of the Fake News Detection and Profiling System. The proposed operational framework, integrating Word embedding and Convolutional Neural Networks (CNNs), is introduced, providing a structured path for model development. A comprehensive research framework is presented, visually depicting the project phases and activities. The detailed description of the framework emphasizes key tasks such as dataset gathering, pre-processing, model development, attribute extraction, and insight visualization. Identified resources, including software and hardware requirements, are essential for the successful implementation of the project. Benchmarks and baselines serve as reference points for evaluating the system's performance, while the emphasis on dataset diversity highlights the critical role of quality data in training the model. The evaluation strategy, featuring metrics like accuracy and precision, ensures a robust assessment of the Word embedding and CNNs model.