



AHMED USAMA SUBHANI
01-134152-003
AHSAN MAQBOOL
01-134152-036

Fake News Detection on Social Media Using Stance Detection

Bachelor of Science in Computer Science

Supervisor: Ms. Momina Moetesum

Department of Computer Science
Bahria University, Islamabad

June 18, 2019

Abstract

Fake news, spread through social media, pose a serious threat to our society. Identifying fake news is a complicated task. The first step leading to an automated fake news detection system is stance detection i.e the relevance between headline and the body of an article. Stance detection can help in identifying click-bait headlines with unrelated body text(a technique mostly used by fake news distributors) and in evaluating the stance different news sources take towards a claim. In this project, we design and develop a system to detect stance between two bodies of text using Natural Language Processing techniques.

Acknowledgments

We are thankful to our respected supervisor Ms. Momina Moetesum who helped us shape this project out in its proper form. From the initiation of this project she was always the source of guidance and supported us on every problem we faced. We also like to thank Ms. Maryam Bibi for her help during the completion of this project.

AHSAN MABOOL & AHMED USAMA SUBHANI
ISLAMABAD, PAKISTAN

April 2019

Contents

1	Introduction	2
1.1	Overview	2
1.2	Problem Description	2
1.3	Project Objective	3
1.4	Project Scope	3
2	Literature Review	4
2.1	Fake News Detection	4
2.2	Stance Detection	4
3	Requirement Specification	6
3.1	Existing System Overview	6
3.2	Proposed System Overview	6
3.3	Requirement Specification	7
3.3.1	Functional Requirements	7
3.3.2	Non-functional Requirements	7
3.4	Use Case	7
3.4.1	Use Case Descriptive Tables	8
4	Design	10
4.1	System Architecture	10
4.2	Design Methodology	11
4.3	High Level Design	12
4.4	GUI Design	13
4.5	Sequence Diagram	14
5	System Implementation	15
5.1	Tools Used	15
5.1.1	Anaconda	15
5.1.2	Scikit-learn	15
5.1.3	Keras	15
5.1.4	Numpy	15
5.1.5	Flask	16
5.2	Methodology	16

5.2.1	Training	16
5.2.2	Testing	17
5.2.3	Deployment	18
5.3	Dataset	19
5.4	Techniques Applied	19
5.4.1	Stance Detection	19
5.4.2	Features	21
5.4.3	Classifier	21
6	System Testing and Evaluation	23
6.1	Graphical User Interface Testing	23
6.2	Usability Testing	23
6.3	Installation Testing	24
6.4	Test Cases	24
6.5	Results	25
7	Conclusions	26
7.1	Future Work	26

List of Figures

3.1	Use Case Diagram	8
4.1	System Architecture	11
4.2	High Level Diagram	12
4.3	GUI Design: Home Page	13
4.4	GUI Design: Results Page	13
4.5	Sequence Diagram	14
5.1	Training	17
5.2	Testing	18
5.3	Dataset Summary	19
5.4	Multi Layer Perceptron	22
6.1	Test Case 1	24
6.2	Test Case 2	24
6.3	Test Case 3	24
6.4	Test Case 4	25
6.5	Confusion Matrix	25

List of Tables

3.1	Use Case 1.	8
3.2	Use Case 2.	9
3.3	Use Case 3.	9
3.4	Use Case 4.	9
3.5	Use Case 5.	9

Acronyms and Abbreviations

TF	Term Frequency
DF	Document Frequency
IDF	Inverse Document Frequency
TF-IDF	Term Frequency Inverse Document Frequency
MLP	Multi Layer Perceptron
NLP	Natural Language Processing
NN	Neural Network
GUI	Graphical User Interface

Chapter 1

Introduction

1.1 Overview

With the advent of social media networks, more and more people have started to rely on news from social media. In contrast to the traditional news mediums, news on social media is easily accessible and less expensive. It is also easier to share and discuss the news with other readers. Despite its advantages, social media is a double-edged sword as it enables “fake news” to be circulated easily as well. Fake news [10], is a made-up story with an intention to deceive and is often used to attract readers to generate ad revenue or to spread propaganda. Fake news spreads confusion among people about current events. Thus, detection of fake news from social media, has become an important issue.

To address this issue, many fact checking services like Snopes [9] and Full-Fact [3] have emerged in the recent past. These services rely on journalists to manually check news authenticity by collecting evidences. However with the amount of news content on the internet rapidly increasing day by day, it is not feasible to classify every news article on the internet manually.

With recent advancements in the field of artificial intelligence technology, researchers believe that the issue of fake news detection can be solved to some extent by using machine learning techniques [4]. However due to the complex nature of human language, detecting fake news accurately is a big challenge. Researchers are trying to fully or partially automate the process of fake news detection using different methods.

1.2 Problem Description

Various social media platforms like Facebook and Twitter, these days, are not only used to interact but also to disseminate different types of news

instances as well. Unfortunately, the massive amount of information flow makes it difficult for moderators to verify the authenticity of each and every news article being posted on their platform. This allows social media users with malicious intentions to spread fake news using these platforms. This maligns the reputation of the said social media platform. Therefore it is important for moderators to verify the authenticity of news instances being posted on their platform in order to control the dissemination of fake news. However checking the authenticity of every item manually is a time-consuming task. An automated tool which can detect fake news using contextual information can help social media moderators to assess the authenticity of news articles being posted on their platforms.

It is observed that fake news distributors use “click-bait” headlines and unrelated body text. Unfortunately, most readers only read the headlines and share them with their friends/followers on social media. By determining the stance between the headline and body, fake news classification can be achieved. In this project, we intend to develop a system which can automatically classify the stance between the headline and the body text and thus determine whether the news article is fake or real.

1.3 Project Objective

The objective of this project is to develop an application to detect fake news by detecting the stance of the news headline with its body text. The system will take a news headline and body as input and will classify stance as one of the following four classes:

- **Agree:** The body text agrees with the headline.
- **Disagree:** The body text disagrees with the headline.
- **Discuss:** The body text discusses the same topic as the headline.
- **Unrelated:** The body text discusses a different topic than the headline.

If the stance is classified as either Unrelated or Disagree, the news is flagged as Fake. Similarly if the stance is either Agree or Discuss, the news is flagged as True.

1.4 Project Scope

This project focuses on textual content from the news articles. Manipulated images/video detection and source based authentication is beyond the scope of this project.

Chapter 2

Literature Review

This chapter reviews the state of art in the field of fake news detection.

2.1 Fake News Detection

Kleinberg and Lefefre [7] describe how they created two datasets for the task of fake news detection and how they used these datasets to train experimental models. The first dataset is obtained using a combination of manual and crowdsourcing annotation efforts. The second dataset is obtained from the web targeting celebrities. Authors used these datasets to conduct analysis to extract linguistic properties present in fake news content. Their fake news detection models achieved accuracies up to 78%. In addition, they also provided a comparative analysis of their models and manual identification of fake news.

Authors in [4] offer a system to determine the authenticity of a news article using a Naive Bayes Classifier. The proposed classifier use BoW features for text classification of body and headlines of news article. The system is trained on Dataset collected by Buzz Feed News. The dataset contains information about Facebook post each of which represent a news articles and implemented as a Software system. There were total 2282 posts. Authors use Spam messages properties to detect given article is a fake or not.

2.2 Stance Detection

Stienberg and and Krejzl [5] describes their participation in the Tweets stance detection task of SemEval 2016. Authors used domain knowledge related features to detect stance of tweets. The dataset used to train their

model contains 78256 tweets and was generated by searching for hashtags on twitter. They defined a domain stance dictionary that lists frequent words in each stance class. Parts-of-speech tags, General Inquirer and entity-centered sentiment dictionaries were applied to extract features. Maximum entropy classifier was used for the classification.

Wu and Cheng [1] proposed a system to to determine the authenticity of news article using Stance detection. System is trained on FNC-1 dataset. The proposed system use different features (i.e. BoW Vectors, Cosine similarity feature, word sentiment etc) and compare there result. This system uses SVM, softmax, multinomial Naive Bayes, and MLP as a classification models.

Chapter 3

Requirement Specification

This chapter includes overview of existing systems and outlines the functional and non-functional requirements for the proposed system.

3.1 Existing System Overview

Most fake news detection projects focus on curating lists of unreliable and questionable websites, flagging a news article on the basis of its source. Some projects apply the techniques of spam detection on news articles. Currently, many schemes have been proposed to automate fake news detection. Most of these schemes have been research based only. According to our knowledge, a user-friendly application has not been developed yet.

3.2 Proposed System Overview

The proposed system is a user-friendly application, allowing the user to input news to check its authenticity. The main purpose of our project is to train a model for fake news detection that can classify the user input. To train the model, Fake News Challenge dataset (FNC-1)[2] is used.

3.3 Requirement Specification

The functional and non-functional requirements of the proposed system are given below.

3.3.1 Functional Requirements

The functional requirements of the proposed system are given below.

- The system must classify the headline-body pair provided by the user.
- The system must display the result to the user.
- The system must display extracted features and the stance of user's input.

3.3.2 Non-functional Requirements

The non-functional requirements of the proposed system are given below.

- The system should be able to handle exceptions and show error messages.
- The source code must be modular, allowing iterative improvements easily.

3.4 Use Case

Figure 3.1 shows the overall use case of the application.

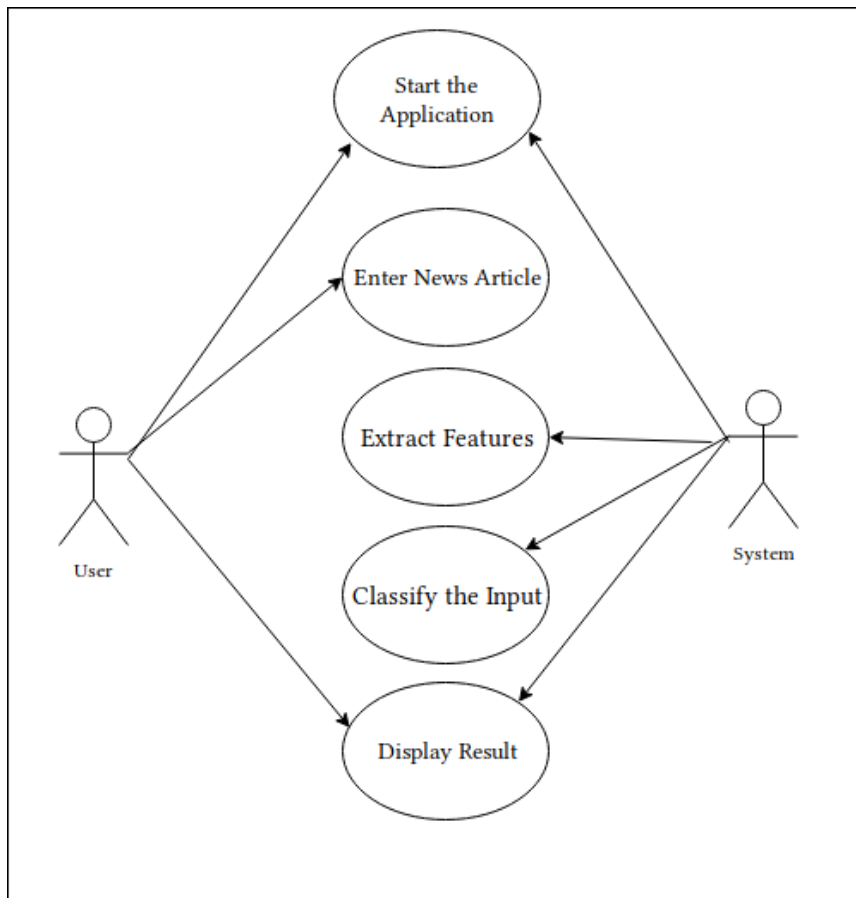


Figure 3.1: Use Case Diagram

3.4.1 Use Case Descriptive Tables

Title: Start the Application

Actors	User, System
Description	User visits the application home page.
Pre-condition	Correct URL is entered.
Post-condition	Home page is displayed.

Table 3.1: Use Case 1.

Title: Enter News Article

Actors	User
Description	User enters headline and body of news article in the form.
Pre-condition	Home page is displayed.
Post-condition	News article is submitted for feature extraction.

Table 3.2: Use Case 2.

Title: Extract Features

Actors	System
Description	TF and IDF weights are calculated from the news article provided.
Pre-condition	User submitted a news article.
Post-condition	Weights are given to the classifier as input.

Table 3.3: Use Case 3.

Title: Classify the Input

Actors	System
Description	Classifier classifies the input.
Pre-condition	Cosine similarity and TF-IDF vectors are extracted.
Post-condition	News article is flagged as either fake or true.

Table 3.4: Use Case 4.

Title: Display Result

Actors	User, System
Description	TF-IDF, class probabilities and final result are displayed.
Pre-condition	Input is classified by the classifier.
Post-condition	User can view the results.

Table 3.5: Use Case 5.

Chapter 4

Design

In this chapter, we will describe the design, modules and interfaces of the project. We provide detailed insight into the inner workings of the system.

4.1 System Architecture

The main function of the application that is deployed at the end of the project is to provide users, with the ability to check the stance between a headline and an article body. To achieve this, we employed machine learning techniques to build a model that can take features of a news article as input and predicts stance as output.

We used a large data-set with headline-body pairs along with their real stance and trained the model on this data. The model determined the relationship between features and stance. Once the relationship is determined, we test the model on new data and compare the predicted results with actual results to determine its accuracy.

The trained model would run at the backend of our application to provide predictions to the user. Our stance detection system has four main components.

- The data-set
- Feature extraction module.
- Classifier
- User Interface

The system architecture of our project is shown in Figure 4.1

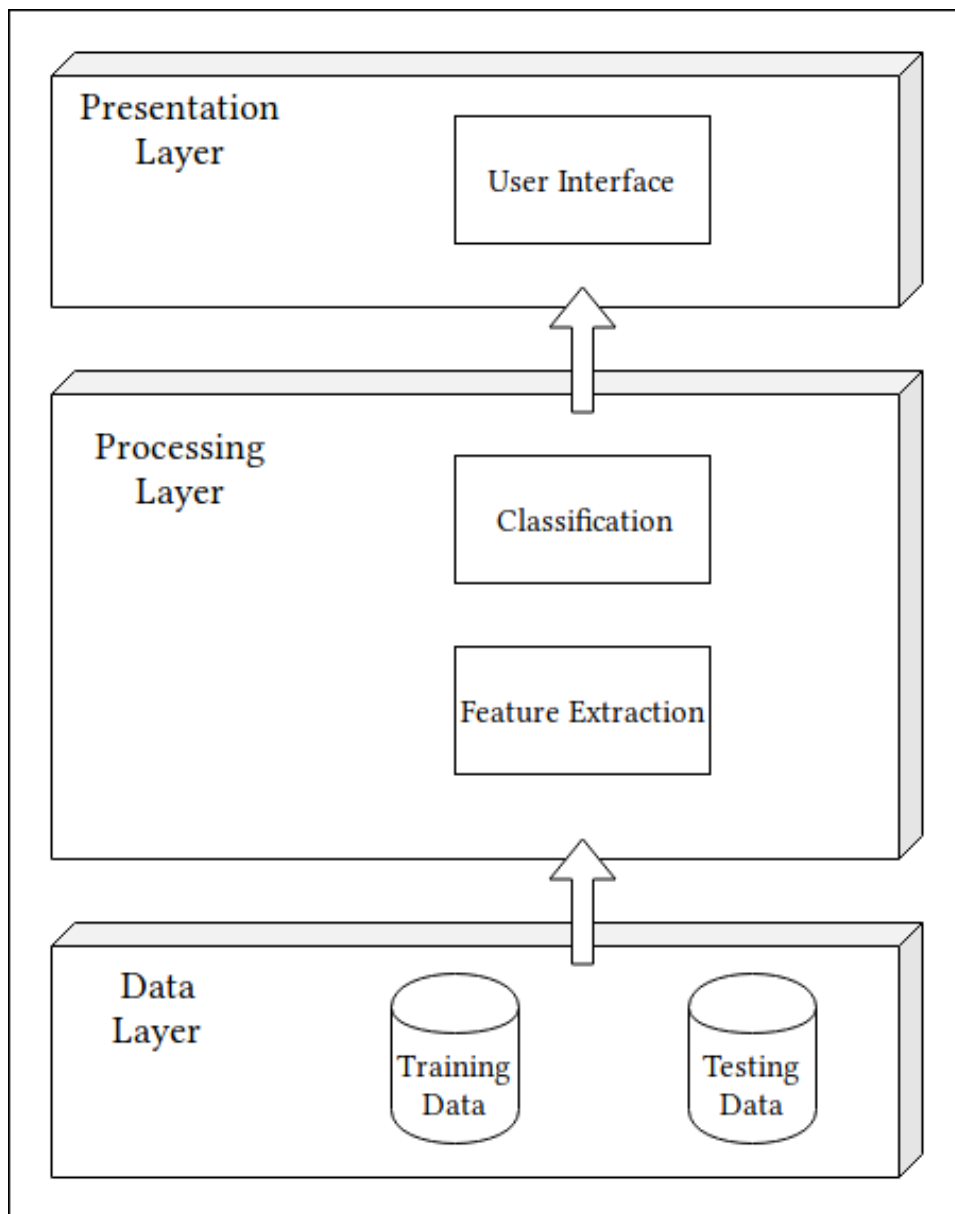


Figure 4.1: System Architecture

4.2 Design Methodology

As the functional requirements of the system are known, an incremental model of the software is used. The incremental model is such which is developed in increments with each increment adding new functionality to the system. A model which is developed in increments with every augmentation adding new functionality and purpose to the system is known as an

incremental model. Each increment contains some addition functionality.

4.3 High Level Design

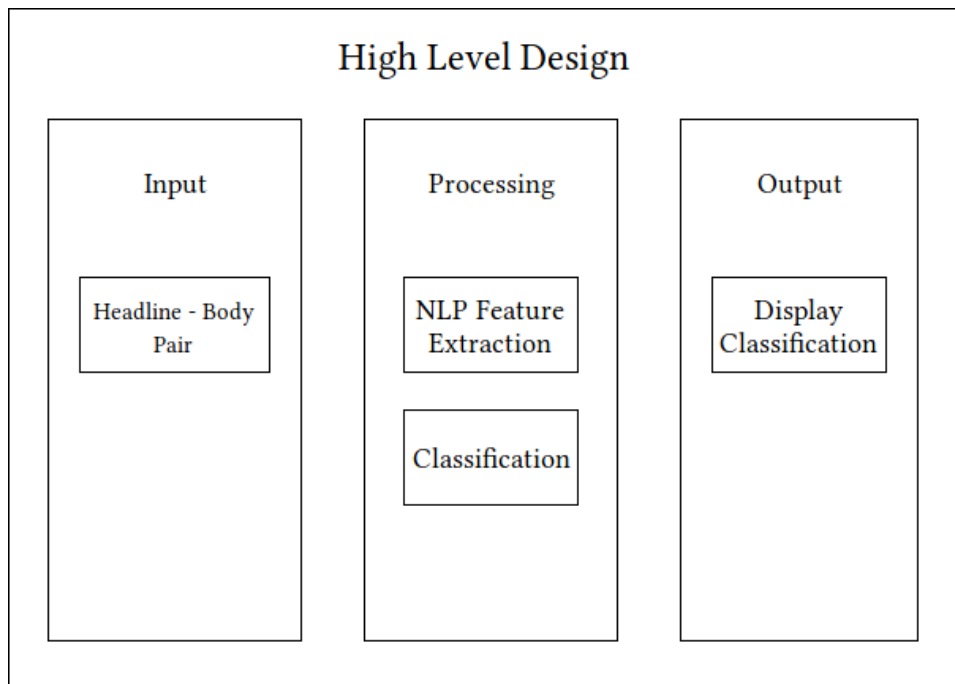
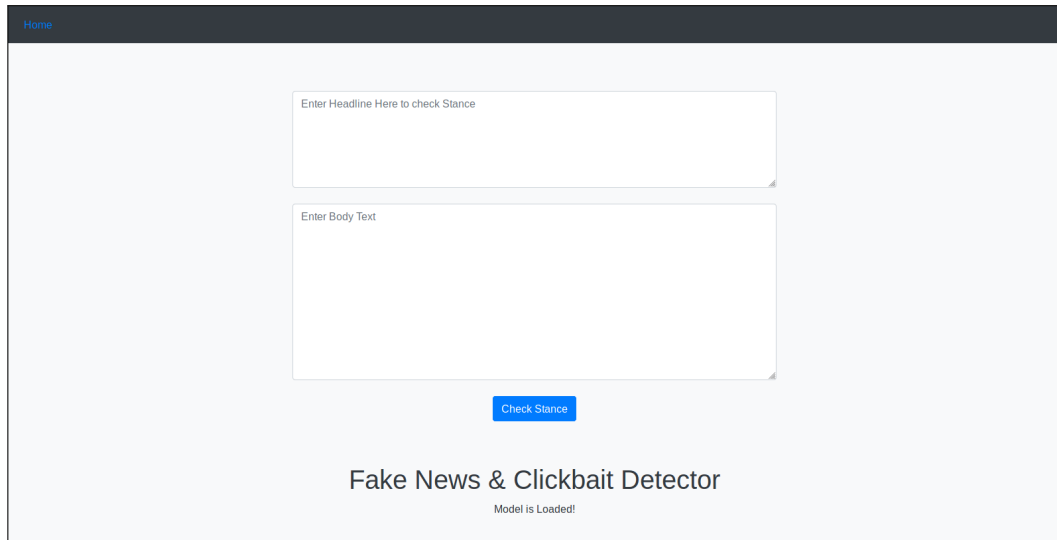


Figure 4.2: High Level Diagram

The high level design is the overall design covering the system functionality. Figure 4.2 gives a high-level view of the system.

4.4 GUI Design

The GUI of our application is simple and self explanatory. The home page as shown in figure 4.3 allows user to enter the input. Clicking the predict button takes user to the results page. The result page as shown in figure 4.4 displays extracted features and the final output.



Home

Enter Headline Here to check Stance

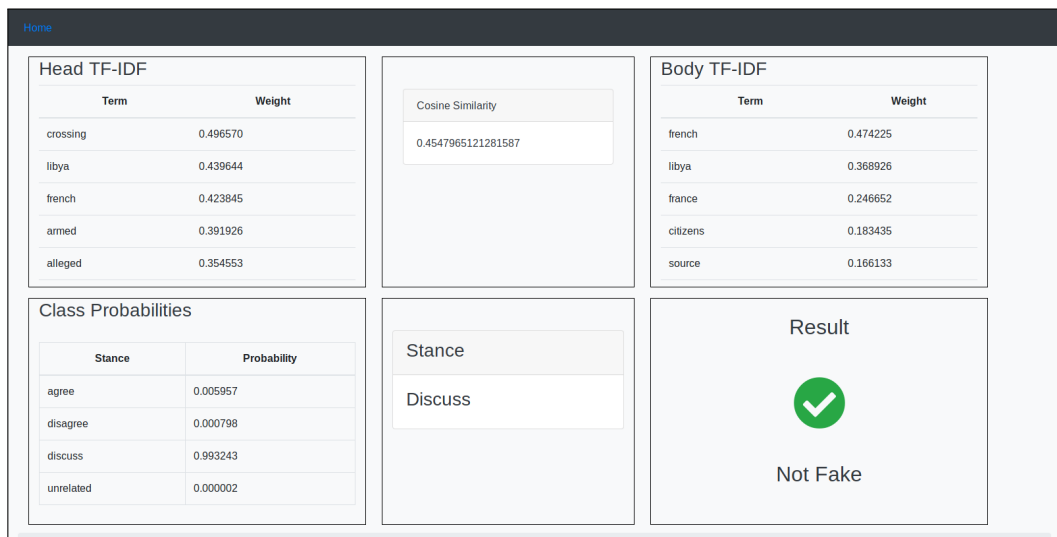
Enter Body Text

Check Stance

Fake News & Clickbait Detector

Model is Loaded!

Figure 4.3: GUI Design: Home Page



Home

Head TF-IDF	
Term	Weight
crossing	0.496570
libya	0.439644
french	0.423845
armed	0.391926
alleged	0.354553

Cosine Similarity

0.4547965121281587


Body TF-IDF	
Term	Weight
french	0.474225
libya	0.368926
france	0.246652
citizens	0.183435
source	0.166133

Class Probabilities	
Stance	Probability
agree	0.005957
disagree	0.000798
discuss	0.993243
unrelated	0.000002

Stance

Discuss

Result



Not Fake

Figure 4.4: GUI Design: Results Page

4.5 Sequence Diagram

A sequence diagram gives the interaction of the objects in a sequence. It depicts the classes involved in the system. The sequence diagram for the working of the system is given in Figure 4.5.

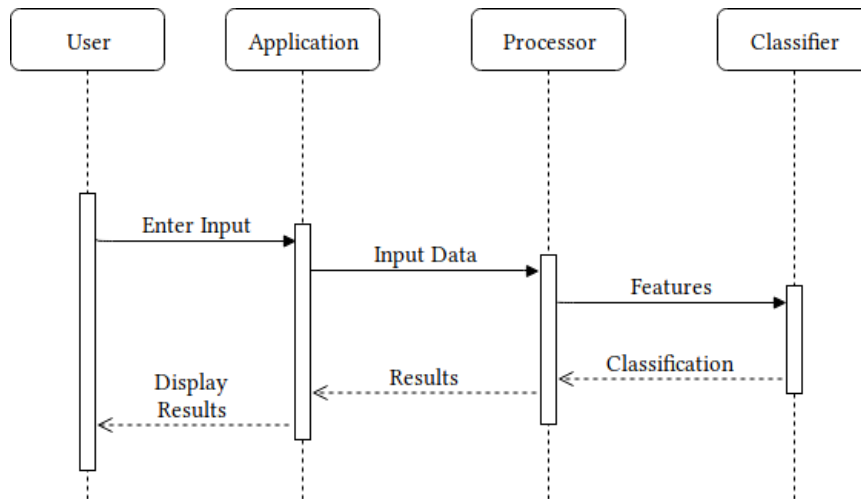


Figure 4.5: Sequence Diagram

Chapter 5

System Implementation

In this chapter we discussed the tools and techniques used in our project.

5.1 Tools Used

Following are the tools used to develop the proposed system.

5.1.1 Anaconda

Anaconda is the most popular platform for Data Science that includes all the widely used tools and packages required for data analysis and manipulation in Python/R. It handles all the overhead of installing packages separately, which can be time-consuming and difficult. The Anaconda Platform will allow us to focus on the problem without worrying about dependencies.

5.1.2 Scikit-learn

Scikit-learn is a machine learning library. We used Scikit-learn to extract features from dataset.

5.1.3 Keras

Keras is a user-friendly neural-network library. It is a high-level API to train deep learning models. We used Keras to train our neural network.

5.1.4 Numpy

Numpy is a Python library that support multi-dimensional matrices and arrays. Numpy is very useful when working with large data-sets.

5.1.5 Flask

Flask is micro web framework for Python. It provides minimal functionality out of the box which can be extended using extensions. We used Flask in our project due to it's simplicity and flexibility.

5.2 Methodology

In this section the methodology to reach the goal is explained.

5.2.1 Training

To train the classification model, following steps are taken.

5.2.1.1 Load Data

In order to train the prediction model, the training data from the data-set provided as CSV files is loaded. Every Instance in the training data has a Headline, Body and Stance.

5.2.1.2 Text to Features

For every instance, Term Frequency and Inverse Document Frequency vectors are extracted from headline and body. Term Frequency-Inverse Document Frequency is calculated for headline and body. Cosine Similarity between TF-IDF of the headline and TF-IDF of the body text is calculated.

A feature vector is prepared for every instance by concatenating TF weight vectors of headline, Cosine Similarity and TF weight vectors of body.

5.2.1.3 Train Classifier

An MLP is trained using feature vectors and stance as its inputs. The trained model is saved for testing and deployment. Figure 5.1 shows the steps involved in training the model.

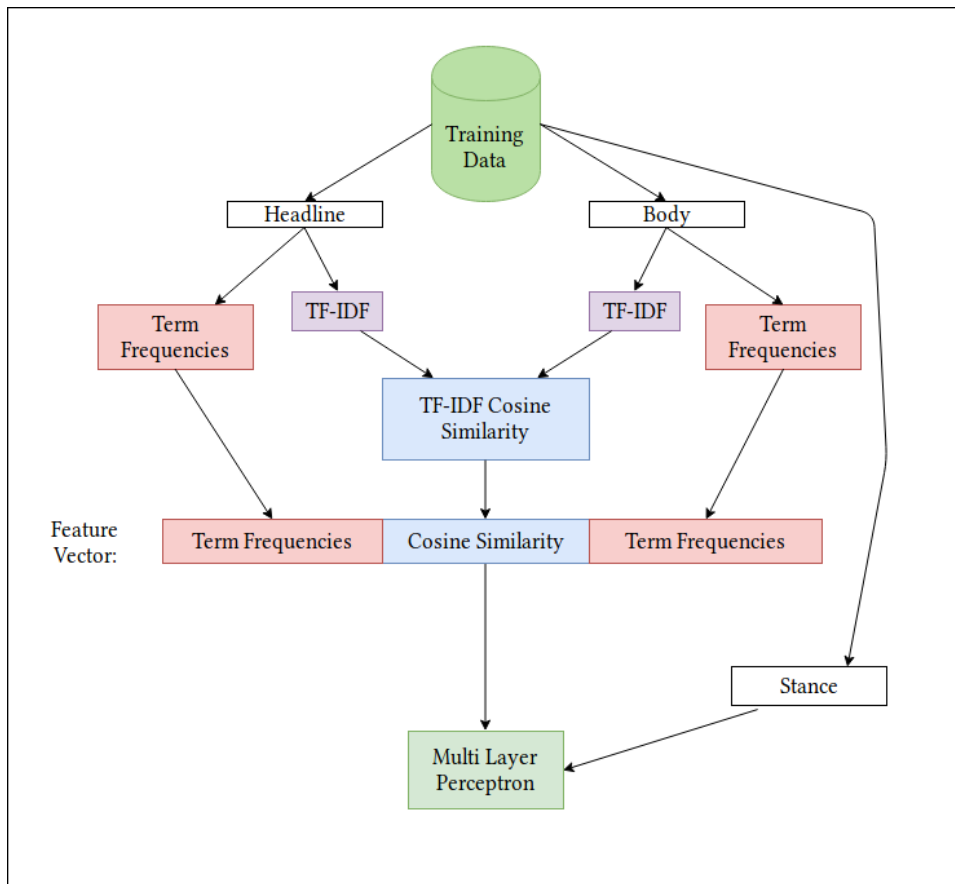


Figure 5.1: Training

5.2.2 Testing

To test the trained model, following steps are taken.

5.2.2.1 Load Test Data

To test the trained model, the test data is loaded from the data-set. Every instance in the test data has a Headline and Body.

5.2.2.2 Text to Features

For every instance, Term Frequency vectors are extracted from headline and body. Term Frequency-Inverse Document Frequency is calculated for headline and body. Cosine Similarity between TF-IDF of the headline and TF-IDF of the body text is calculated. Term Frequencies of headline, Cosine Similarity and Term Frequency of body are concatenated in a feature vector.

5.2.2.3 Test Classifier

Trained model is loaded and predictions are made using feature vector as the model's input. Confusion matrix and accuracy are calculated and displayed. Figure 5.2 shows the steps involved in testing the model's accuracy.

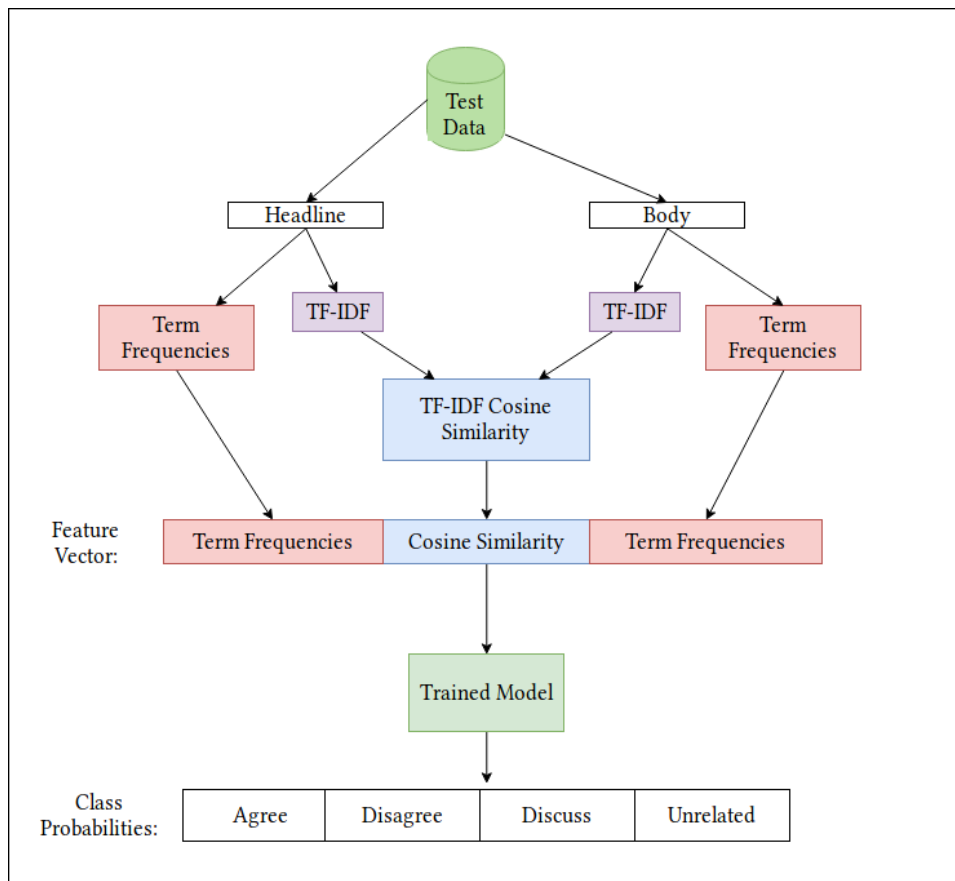


Figure 5.2: Testing

5.2.3 Deployment

To allow users to use our system for stance detection, we created a web application using Flask framework. The front-end is designed in HTML/Bootstrap

and provides a simple interface to give inputs and view results. The model is loaded when the user starts the application. After taking a headline and body from user as input, extracted features and the stance is shown to the user.

5.3 Dataset

The data-set we used to train our model is called FNC-1 [2]. It consists of 49972 headline-body pairs with stances labeled by expert journalists. The test data consists of 5025 headline-body pairs. Detailed data statistics is given in Figure 5.3

# Headline-body pairs	49972		
# Headlines	1648		
# Bodies	1683		
# Bodies in test set	169		
# Headline-body pairs in test set	5025		
Average # tokens of headline	12.6		
Average # tokens of body	427.5		
<i>Unrelated</i>	<i>Discuss</i>	<i>Agree</i>	<i>Disagree</i>
73.1%	17.8%	7.4%	1.7%

Figure 5.3: Dataset Summary

5.4 Techniques Applied

Techniques applied in order to implement the algorithm proposed are as under:

5.4.1 Stance Detection

The goal of stance detection is to predict whether the given body text is related to the given headline, and what is the relation between them.

Input

A headline and body text, either from same article or from different articles.

Output

Classify the stance as one of the following:

- **Agree**
- **Disagree**
- **Discuss**
- **Unrelated**

5.4.2 Features

We used two features: Term Frequency and term frequency-inverse document frequency[8].

5.4.2.1 Term Frequency

Term Frequency measures how frequently a word appear in a document. Since every document is different in length, it is possible that a term would appear more times in long documents than in shorter documents. TF score is determined by dividing word occurrences by the document length.

$$TF(t) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}}$$

5.4.2.2 Inverse Document Frequency

IDF of a word is the measure of how significant that word is in the dataset i.e The more documents a word appears in, the less valuable that word is to differentiate any given document.

$$IDF(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents with term } t}\right)$$

5.4.2.3 TF-IDF

TF-IDF is used to extract most important terms in an article.

$$tfidf(t) = TF(t) \times IDF(t)$$

A feature vector is generated by concatenating the following:

- TF vector of headline,
- TF vector of body,
- Cosine Similarity between TF-IDF of headline and body.

5.4.3 Classifier

The classifier we used is a Multi Layer Perceptron [6] with one hidden layer of 100 units and a softmax layer for output. The classifier predicts the probabilities of classes ('agree', 'disagree', 'discuss', 'unrelated'). It is implemented using Keras with Tensorflow backend. Figure 5.4 shows an overview of the classification model.

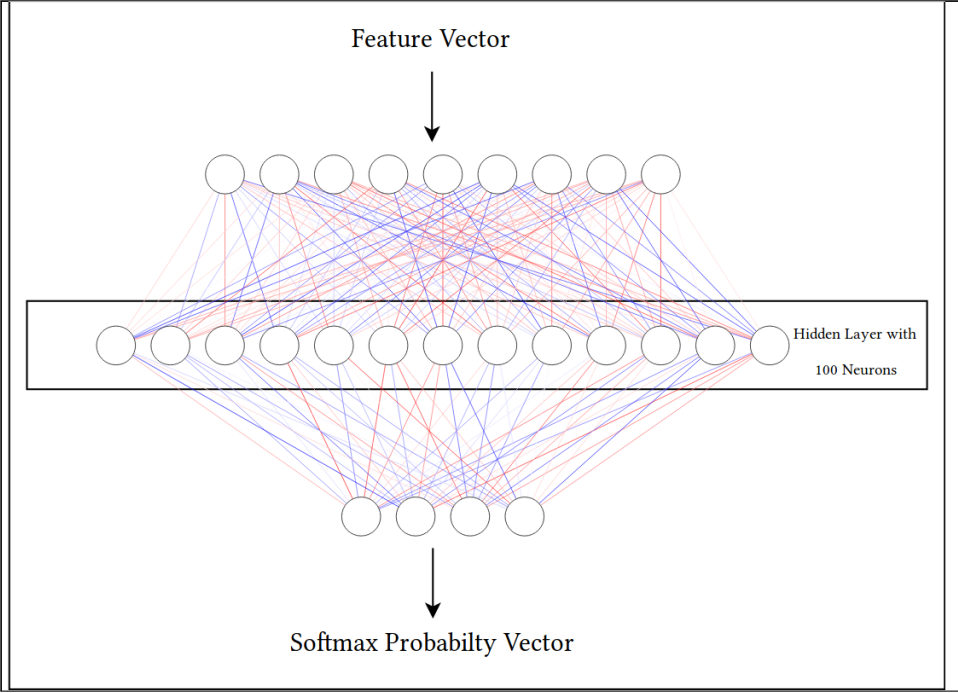


Figure 5.4: Multi Layer Perceptron

Chapter 6

System Testing and Evaluation

This chapter depicts various testing methods which we conducted on our application to test its functionality , to be used for the validity of our system. In software development process the most important phase is testing as it pin points errors, problems and mistakes in software.

6.1 Graphical User Interface Testing

Graphical user interface is one of the most important parts of a respective product or system. Through a GUI, user can communicate through out the system. GUI guides how to use the systems and makes it easy to understand. A GUI should be appealing, full of interactiveness, and simple in accordance to the user as ease of use should be experienced by user. Our project GUI is simple, easy to use and self explanatory. Users who are not familiar to our application were given the application to test for ease of use. They were able to understand and use the system with ease. All of them had no issue during the testing.

6.2 Usability Testing

The usability testing is the testing which is done by the actual users of the system. As mentioned in the GUI testing, users were able to use our system with ease. It is the type of testing in which the actual users of the application test it against standards. As mentioned earlier all users were able to use our application easily with no issue.

6.3 Installation Testing

This test is performed to ensure our system can be used across multiple platforms. As our project is web based, users can access it through a web browser without installing anything.

6.4 Test Cases

The test cases for our system are mentioned in this section.

Test-Case ID	TC 1
Description	Classify the input
Requirements	User must have to give input
Steps to be taken	Copy Headline and Body of news article in Headline and Body section and press Predict button
Expected Results	Correct output is displayed
Actual Results	Correct output and class probabilities shown
Status	Success

Figure 6.1: Test Case 1

Test-Case ID	TC 2
Description	Extract TF-IDF Features
Requirements	User must have to give input
Steps to be Taken	Copy Headline and Body of news article in Headline and Body section and press Predict button
Expected Results	TF-IDF Features will be displayed
Actual Results	TF-IDF Features is displayed
Status	Success

Figure 6.2: Test Case 2

Test-Case ID	TC 3
Description	Run application on different browser
Requirements	All browser are up to date
Steps to be Taken	Run application on each browser (i.e. FF,Chrome)
Expected Results	Application will be running on every browser
Actual Results	Application is running on every browser
Status	Success

Figure 6.3: Test Case 3

Test-Case ID	TC 4
Description	To Start application properly
Requirements	When user enter the URL, it should launch
Steps to be Taken	Correct URL should be entered
Expected Results	Application will instigate without crashing
Actual Results	Application Launched successfully
Status	Success

Figure 6.4: Test Case 4

6.5 Results

The performance of our system is summarized by the confusion matrix in Table 6.5.

Predicted True	Agree	Disagree	Discuss	Unrelated	Accuracy
Agree	336	0	274	87	48%
Disagree	99	188	248	162	27%
Discuss	96	1	526	74	76%
Unrelated	14	1	40	642	92%

Figure 6.5: Confusion Matrix

Chapter 7

Conclusions

Fake news is a serious problem today and this motivated us to solve this problem by working on this project. During this project we encountered many challenges but overall it was a great educational and learning experience. It also paved way for us to learn about machine learning and natural language processing.

7.1 Future Work

Although the accuracy achieved by our project is satisfactory, for better results, accuracy for the classes 'agree' and 'disagree' can be improved by collecting more data for these classes and adding it to the training data.

References

- [1] Xiaowei Wu Sizhu Cheng Zixian Chai. “Fake News Stance Detection”. In: (2016).
- [2] FakeNewsChallenge. *FakeNewsChallenge/fnc-1*. June 2017. URL: <https://github.com/FakeNewsChallenge/fnc-1/>.
- [3] *Fullfact.org*. URL: <https://www.fullfact.org/>.
- [4] Mykhailo Granik and Volodymyr Mesyura. “Fake news detection using naive Bayes classifier”. In: *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*. IEEE. 2017, pp. 900–903.
- [5] Peter Krejzl and Josef Steinberger. “UWB at SemEval-2016 task 6: stance detection”. In: (2016), pp. 408–412.
- [6] Sankar K Pal and Sushmita Mitra. “Multilayer perceptron, fuzzy sets, and classification”. In: *IEEE Transactions on neural networks* 3.5 (1992), pp. 683–697.
- [7] Verónica Pérez-Rosas et al. “Automatic detection of fake news”. In: *arXiv preprint arXiv:1708.07104* (2017).
- [8] Juan Ramos et al. “Using tf-idf to determine word relevance in document queries”. In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. Piscataway, NJ. 2003, pp. 133–142.
- [9] *Snopes.com*. URL: <https://www.snopes.com/>.
- [10] Sabrina Tavernise. “As fake news spreads lies, more readers shrug at the truth”. In: *New York Times* 6 (2016).