FAKE DETECTOR: EFFECTIVE FAKE NEWS DETECTION WITHDEEP DIFFUSIVE NEURAL NETWORK

A Project Report submitted

in partial fulfillment of

the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY
in
INFORMATION TECHNOLOGY

Submitted By

P. SURYA SARATH BHARADWAJ	**: 17WJ1A1236**
PALADUGU SAI VINEET	**: 17WJ1A1238**
PONNA AKHIL	**: 17WJ1A1241**

Under the Guidance of
P. NARESH
Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY
GURU NANAK INSTITUTIONS TECHNICAL CAMPUS (Autonomous)
(Affiliated to JNTUH, Accredited by NBA)
Ibrahimpatnam, R. R. District, Telangana - 501506.

(2017-2021)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that this project report entitled **"Fake Detector: Effective Fake News Detection with Deep Diffusive Neural Network"** submitted by **P. Surya Sarath Bharadwaj (17WJ1A1236), Paladugu Sai Vineet (17WJ1A1238), Ponna Akhil (17WJ1A1241)** in partial fulfilment for the award of the degree of Bachelor of Technology in Information Technology prescribed by Guru Nanak Institutions Technical Campus (Autonomous) affiliated to Jawaharlal Nehru Technological University, Hyderabad during the academic year 2020-2021.

Internal Guide	**Project Coordinator**	**HOD**	**Associate Director**
Mr. P. Naresh	Mrs. P. Padma	Dr. M.I. Thariq Hussan	Dr. Rishi Sayal

External Examiner

Vision of the Institution

To be a world class educational and research institution in the service of humanity by promoting high quality Engineering, Management and Pharmacy education.

Mission of the Institution

M1: Imbibe soft skills and technical skills.

M2: Develop the faculty to reach international standards.

M3: Maintain high academic standards and teaching quality that promotes analytical thinking and independent judgement.

M4: Promote research, innovation and product development collaboration with reputed foreign universities.

M5: Offer collaborative industry program in the emerging areas and spirit of enterprise.

Vision of the Department

To be a premier Information Technology department in the region by providing high quality education.

Mission of the Department

M1: Nurture young individuals into knowledgeable, skillful and ethical professionals in their pursuit of Information Technology.

M2: Transform the students through excellent teaching learning process and sustain high performance by innovations.

M3: Extensive partnerships and collaborations with foreign universities.

M4: Develop industry-interaction for innovation and product development.

DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: Produce industry ready graduates having the ability to apply academic knowledge across the disciplines and in emerging areas of Information Technology for higher studies, research, employability, product development and handle the realistic problems.

PEO 2: Graduates will have good communication skills, possess ethical conduct, sense of responsibility to serve the society and protect the environment.

PEO 3: Graduates will have excellence in soft skills, managerial skills, leadership qualities and understand the need for lifelong learning for a successful professional career.

PROGRAMME OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

1. Ability to design, develop, test and debug software applications, evaluate and recognize potential risks and provide innovative solutions.

2. Explore technical knowledge in diverse areas of Information Technology for upliftment of society, successful career, entrepreneurship and higher studies.

ACKNOWLEDGEMENT

We wish to express our sincere thanks to Chairman **Sardar T.S. Kohli** and Vice-Chairman **Sardar G.S. Kohli** of **Guru Nanak Institutions** for providing us all necessary facilities, re-sources and infrastructure for completing this project work.

We express a whole hearted gratitude to our Managing Director **Dr. H.S. Saini** for providing strong vision in engineering education through accreditation policies under regular training in upgrading education for the faculty members.

We express a whole hearted gratitude to our Director **Dr. M. Ramalinga Reddy** for providing us the constructive platform to launch our ideas in the area of Information Technology and improving our academic standards.

We express a whole hearted gratitude to our Associate Director **Dr. Rishi Sayal** for providing us the conducive environment for carrying through our academic schedules and projects with ease.

We have been truly blessed to have a wonderful advisor **Dr. M.I. Thariq Hussan** HOD Department of Information Technology for guiding us to explore the ramification of our work and we express our sincere gratitude towards him for leading me throughout the project work.

We specially thank our internal guide **Mr. P. Naresh** Assistant Professor of Information Technology for his valuable suggestions and constant guidance in every stage of the project.

We express our sincere thanks to all the faculties of Information Technology department who helped us in every stage of our project by providing their valuable suggestions and support.

DECLARATION

We **P. Surya Sarath Bharadwaj (17WJ1A1236), Paladugu Sai Vineet (17WJ1A1238), Ponna Akhil (17WJ1A1241)** hereby declare that the project report entitled "**Fake Detector: Effective Fake News Detection with Deep Diffusive Neural Network**" under the esteemed guidance of **Mr. P. Naresh**, Assistant Professor of Information Technology submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Information Technology. This is a record of bona fide work carried out by us and the results embodied in this project report have not been submitted to any other University or Institute for the award of Degree or Diploma.

Date:

Place: GNITC

P. SURYA SARATH BHARADWAJ : 17WJ1A1236
PALADUGU SAI VINEET : 17WJ1A1238
PONNA AKHIL : 17WJ1A1241

ABSTRACT

In recent years, due to the booming development of online social networks, fake news for various commercial and political purposes has been appearing in large numbers and widespread in the online world. With deceptive words, online social network users can get infected by this online fake news easily, which has brought about tremendous effects on the offline society already. An important goal in improving the trustworthiness of information in online social networks is to identify the fake news timely.

This paper aims at investigating the principles, methodologies and algorithms for detecting fake news articles, creators and subjects from online social networks and evaluating the corresponding performance. This paper addresses the challenges introduced by the unknown characteristics of fake news and diverse connections among news articles, creators and subjects. This paper introduces a novel gated graph neural network, namely FAKEDETECTOR. Based on a set of explicit and latent features extracted from the textual information, FAKEDETECTOR builds a deep diffusive network model to learn the representations of news articles, creators and subjects simultaneously. Extensive experiments have been done on a real-world fake news dataset to compare FAKEDETECTOR with several state-of-the-art models, and the experimental results are provided in the full-version of this paper.

TABLE OF CONTENTS

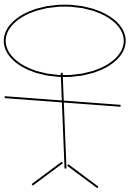
CHAPTER NO.	TITLE	PAGE NO.
1.	**CHAPTER 1: INTRODUCTION**	1 – 24
	1.1 GENERAL	1
	1.2 SCOPE OF THE PROJECT	16
	1.3 OBJECTIVE	17
	1.4 PROBLEM STATEMENT	17
	1.5 EXISTING SYSTEM	18
	1.5.1 EXISTING SYSTEM DISADVANTAGES	18
	1.5.2 LITERATURE SURVEY	19
	1.6 PROPOSED SYSTEM	24
	1.6.1 PROPOSED SYSTEM ADVANTAGES	24
2.	**CHAPTER 2: PROJECT DESCRIPTION**	25 – 30
	2.1 GENERAL	25
	2.2 METHODOLOGIES	25
	2.2.1 MODULES NAME	25
	2.2.2 MODULES EXPLANATION	25
	2.2.3 GIVEN INPUT AND EXPECTED OUTPUT	28
3.	**CHAPTER 3: REQUIREMENTS**	31 – 33
	3.1 GENERAL	31
	3.2 HARDWARE REQUIREMENTS	31
	3.3 SOFTWARE REQUIREMENTS	32
	3.4 FUNCTIONAL SPECIFICATION	32
	3.5 NON-FUNCTIONAL SPECIFICATION	33
4.	**CHAPTER 4: SYSTEM DESIGN**	34 – 39
	4.1 GENERAL	34
	4.1.1 USE CASE DIAGRAM	34
	4.1.2 CLASS DIAGRAM	35
	4.1.3 SEQUENCED DIAGRAM	36
	4.1.4 ACTIVITY DIAGRAM	37
	4.1.5 DATA FLOW DIAGRAM	38
	4.2 SYSTEM ARCHITECTURE	39
5.	**CHAPTER 5: SOFTWARE SPECIFICATION**	40 – 53
	5.1 GENERAL	40

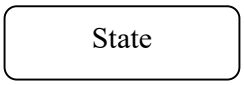
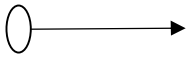
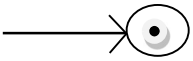
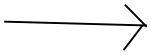
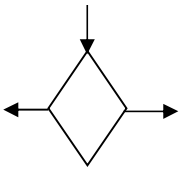
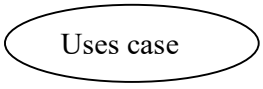
6.	**CHAPTER 6: IMPLEMENTATION** 6.1 GENERAL 6.2 IMPLEMENTATION	54 – 75 54 54
7.	**CHAPTER 7: SNAPSHOTS** 7.1 GENERAL 7.2 VARIOUS SNAPSHOTS	76 – 84 76 76
8.	**CHAPTER 8: SOFTWARE TESTING** 8.1 GENERAL 8.2 DEVELOPING METHODOLOGIES 8.3 TYPES OF TESTING 8.3.1 UNIT TESTING 8.3.2 FUNCTIONAL TEST 8.3.3 SYSTEM TEST 8.3.4 PERFORMANCE TEST 8.3.5 INTEGRATION TESTING 8.3.6 ACCEPTANCE TESTING 8.3.7 BUILD THE TEST PLAN 8.4 UNIT TESTING 8.5 INTEGRATION TEST 8.6 ACCEPTANCE TESTING	85 – 88 85 85 85 85 86 86 86 86 87 87 87 88 88
9.	**CHAPTER 9: APPLICATIONS AND FUTURE ENHANCEMENT** 9.1 APPLICATIONS 9.2 FUTURE ENHANCEMENTS	89 89 89
10.	**CHAPTER 10: CONCLUSION AND REFERENCES** 10.1 CONCLUSION 10.2 REFERENCES	90 – 91 90 91

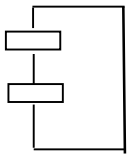
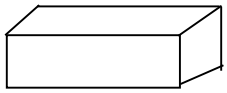
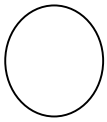
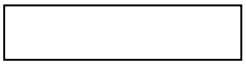
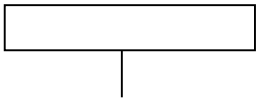
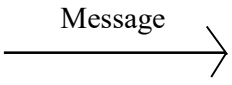
LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO.
4.1.1	Use case Diagram	35
4.1.2	Class Diagram	12
4.1.3	Sequence Diagram	13
4.1.4	Activity Diagram	14
4.1.5	Data flow Diagram	15
4.2	System Architecture	16

LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class	*+ public* *-private* *# protected* *Class Name* *-attribute* *-attribute* *+operation* *+operation* *+operation*	Represents a collection of similar entities grouped.
2.	Association	NAME Class A — Class B Class A — Class B	Associations represent static relationships between classes. Roles represent the way the two classes see each other.
3.	Actor		It aggregates several classes into a single class.
4.	Aggregation	Class A Class B Class A Class B	Interaction between the system and external environment

5.	Relation (uses)	uses	Used for additional process communication.
6.	Relation (extends)	extends	Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the processes.
9.	Initial State		The initial state of the object
10.	Final state		The final state of the object
11.	Control flow		Represents various control flows between the states.
12.	Decision box		Represents decision-making process from a constraint
13.	Use case		Interact ion between the system and external environment.

14.	Component		Represents physical modules which are a collection of components.
15.	Node		Represents physical modules which are a collection of components.
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
17.	External entity		Represents external entities such as keyboard,sensors,etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message		Represents the message exchanged.

LIST OF ABBREVATION

S.NO	ABBREVATION	EXPANSION
1.	DB	Data Base
2.	ML	Machine Learning
3.	SL	Supervised Learning
4.	AI	Artificial intelligence
5.	ANNs	Artificial Neural Networks
6.	RF	Random Forest

CHAPTER 1
INTRODUCTION

1.1 GENERAL

Fake news denotes a type of yellow press which intentionally presents misinformation or hoaxes spreading through both traditional print news media and recent online social media. Fake news has been existing for a long time, since the "Great moon hoax" published in 1835. In recent years, due to the booming developments of online social networks, fake news for various commercial and political purposes has been appearing in large numbers and widespread in the online world. With deceptive words, online social network users can get infected by this online fake news easily, which has brought about tremendous effects on the offline society already. During the 2016 US president election, various kinds of fake news about the candidates widely spread in the online social networks, which may have a significant effect on the election results. According to a post-election statistical report, online social networks account for more than 41.8% of the fake news data traffic in the election, which is much greater than the data traffic shares of both traditional TV/radio/print medium and online search engines respectively. An important goal in improving the trustworthiness of information in online social networks is to identify the fake news timely, which will be the main task studied in this paper.

Fake news has significant differences compared with traditional suspicious information, like spams, in various aspects:

(1) impact on society: spams usually exist in personal emails or specific review websites and merely have a local impact on a small number of audiences, while the impact fake news in online social networks can be tremendous due to the massive user numbers globally, which is further boosted by the extensive information sharing and propagation among these users.

(2) audiences' initiative: instead of receiving spam emails passively, users in online social networks may seek for, receive and share news information actively with no sense about its correctness; and

(3) identification difficulty: via comparisons with abundant regular messages (in emails or review websites), spams are usually easier to be distinguished, whereas identifying fake news with erroneous information is incredibly challenging, since it requires both tedious evidence-collecting and careful fact-checking due to the lack of other comparative news articles available.

These characteristics aforementioned of fake news pose new challenges on the detection task. Besides detecting fake news articles, identifying the fake news creators and subjects will actually be more important, which will help completely eradicate a large number of fake news from the origins in online social networks. Generally, for the news creators, besides the articles written by them, we are also able to retrieve his/her profile information from either the social network website or external knowledge libraries, e.g., Wikipedia or government-internal database, which will provide fundamental complementary information for his/her background check. Meanwhile, for the news subjects, we can also obtain its textual descriptions or other related information, which can be used as the foundations for news subject credibility inference.

From a higher-level perspective, the tasks of fake news article, creator and subject detection are highly correlated, since the articles written from a trustworthy person should have a higher credibility, while the person who frequently posting unauthentic information will have a lower credibility on the other hand. Similar correlations can also be observed between news articles and news subjects. In the following part of this paper, without clear specifications, we will use the general fake news term to denote the fake news articles, creators and subjects by default.

Deep Learning:

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the back propagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional

nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning. Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input. Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification.

Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned.

For classification tasks, higher layers of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations. An image, for example, comes in the form of an array of pixel values, and the learned features in the first layer of representation typically represent the presence or absence of edges at particular orientations and locations in the image. The second layer typically detects motifs by spotting particular arrangements of edges, regardless of small variations in the edge positions. The third layer may assemble motifs into larger combinations that correspond to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts. The key aspect of deep

learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure. Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition and speech recognition, it has beaten other machine-learning techniques at predicting the activity of potential drug molecules, analysing particle accelerator data, reconstructing brain circuits, and predicting the effects of mutations in non-coding DNA on gene expression and disease.

Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding, particularly topic classification, sentiment analysis, question answering and language translation. We think that deep learning will have many more successes in the near future because it requires very little engineering by hand, so it can easily take advantage of increases in the amount of available computation and data. New learning algorithms and architectures that are currently being developed for deep neural networks will only accelerate this progress.

Supervised learning:

The most common form of machine learning, deep or not, is supervised learning. Imagine that we want to build a system that can classify images as containing, say, a house, a car, a person or a pet. We first collect a large data set of images of houses, cars, people and pets, each labelled with its category. During training, the machine is shown an image and produces an output in the form of a vector of scores, one for each category. We want the desired category to have the highest score of all categories, but this is unlikely to happen before training. We compute an objective function that measures the error (or distance) between the output scores and the desired pattern of scores. The machine then modifies its internal adjustable parameters to reduce this error. These adjustable parameters, often called weights, are real numbers that can be seen as ‘knobs’ that define the input–output function of the machine.

In a typical deep-learning system, there may be hundreds of millions of these adjustable weights, and hundreds of millions of labelled examples with which to train the machine. To

properly adjust the weight vector, the learning algorithm computes a gradient vector that, for each weight, indicates by what amount the error would increase or decrease if the weight were increased by a tiny amount. The weight vector is then adjusted in the opposite direction to the gradient vector. The objective function, averaged over all the training examples, can be seen as a kind of hilly landscape in the high-dimensional space of weight values. The negative gradient vector indicates the direction of steepest descent in this landscape, taking it closer to a minimum, where the output error is low on average. In practice, most practitioners use a procedure called stochastic gradient descent (SGD). This consists of showing the input vector for a few examples, computing the outputs and the errors, computing the average gradient for those examples, and adjusting the weights accordingly. The process is repeated for many small sets of examples from the training set until the average of the objective function stops decreasing. It is called stochastic because each small set of examples gives a noisy estimate of the average gradient over all examples. This simple procedure usually finds a good set of weights surprisingly quickly when compared with far more elaborate optimization techniques. After training, the performance of the system is measured on a different set of examples called a test set. This serves to test the generalization ability of the machine — its ability to produce sensible answers on new inputs that it has never seen during training.

Many of the current practical applications of machine learning use linear classifiers on top of hand-engineered features. A two-class linear classifier computes a weighted sum of the feature vector components. If the weighted sum is above a threshold, the input is classified as belonging to a particular category. Since the 1960s we have known that linear classifiers can only carve their input space into very simple regions, namely half-spaces separated by a hyperplane. But problems such as image and speech recognition require the input–output function to be insensitive to irrelevant variations of the input, such as variations in position, orientation or illumination of an object, or variations in the pitch or accent of speech, while being very sensitive to particular minute variations (for example, the difference between a white wolf and a breed of wolf-like white dog called a Samoyed). At the pixel level, images of two Samoyeds in different poses and in different environments may be very different from each other, whereas two images of a Samoyed and a wolf in the same position and on similar

backgrounds may be very similar to each other. A linear classifier, or any other 'shallow' classifier operating on raw pixels could not possibly distinguish the latter two, while putting the former two in the same category. This is why shallow classifiers require a good feature extractor that solves the selectivity–invariance dilemma — one that produces representations that are selective to the aspects of the image that are important for discrimination, but that are invariant to irrelevant aspects such as the pose of the animal. To make classifiers more powerful, one can use generic non-linear features, as with kernel methods, but generic features such as those arising with the Gaussian kernel do not allow the learner to generalize well far from the training examples. The conventional option is to hand design good feature extractors, which requires a considerable amount of engineering skill and domain expertise. But this can all be avoided if good features can be learned automatically using a general-purpose learning procedure. This is the key advantage of deep learning. A deep-learning architecture is a multilayer stack of simple modules, all (or most) of which are subject to learning, and many of which compute non-linear input–output mappings. Each module in the stack transforms its input to increase both the selectivity and the invariance of the representation. With multiple non-linear layers, say a depth of 5 to 20, a system can implement extremely intricate functions of its inputs that are simultaneously sensitive to minute details — distinguishing Samoyeds from white wolves — and insensitive to large irrelevant variations such as the background, pose, lighting and surrounding objects.

Backpropagation to train multilayer architectures:

From the earliest days of pattern recognition, the aim of researchers has been to replace hand-engineered features with trainable multilayer networks, but despite its simplicity, the solution was not widely understood until the mid-1980s. As it turns out, multilayer architectures can be trained by simple stochastic gradient descent. As long as the modules are relatively smooth functions of their inputs and of their internal weights, one can compute gradients using the backpropagation procedure. The idea that this could be done, and that it worked, was discovered independently by several different groups during the 1970s and 1980s. The backpropagation procedure to compute the gradient of an objective function with respect to the weights of a multilayer stack of modules is nothing more than a practical

application of the chain rule for derivatives. The key insight is that the derivative (or gradient) of the objective with respect to the input of a module can be computed by working backwards from the gradient with respect to the output of that module (or the input of the subsequent module). The back propagation equation can be applied repeatedly to propagate gradients through all modules, starting from the output at the top (where the network produces its prediction) all the way to the bottom (where the external input is fed). Once these gradients have been computed, it is straightforward to compute the gradients with respect to the weights of each module. Many applications of deep learning use feedforward neural network architectures (Fig. 1), which learn to map a fixed-size input (for example, an image) to a fixed-size output (for example, a probability for each of several categories).

To go from one layer to the next, a set of units compute a weighted sum of their inputs from the previous layer and pass the result through a non-linear function. At present, the most popular non-linear function is the rectified linear unit (ReLU), which is simply the half-wave rectifier $f(z)= \max(z, 0)$. In past decades, neural nets used smoother non-linearities, such as $\tanh(z)$ or $1/(1+\exp(-z))$, but the ReLU typically learns much faster in networks with many layers, allowing training of a deep supervised network without unsupervised pre-training. Units that are not in the input or output layer are conventionally called hidden units. The hidden layers can be seen as distorting the input in a non-linear way so that categories become linearly separable by the last layer (Fig. 1).

In the late 1990s, neural nets and backpropagation were largely forsaken by the machine-learning community and ignored by the computer-vision and speech-recognition communities. It was widely thought that learning useful, multistage, feature extractors with little prior knowledge was infeasible. In particular, it was commonly thought that simple gradient descent would get trapped in poor local minima — weight configurations for which no small change would reduce the average error. In practice, poor local minima are rarely a problem with large networks. Regardless of the initial conditions, the system nearly always reaches solutions of very similar quality.

Recent theoretical and empirical results strongly suggest that local minima are not a serious issue in general. Instead, the landscape is packed with a combinatorially large number of saddle points where the gradient is zero, and the surface curves up in most dimensions and

curves down in the remainder. The analysis seems to show that saddle points with only a few downward curving directions are present in very large numbers, but almost all of them have very similar values of the objective function. Hence, it does not much matter which of these saddle points the algorithm gets stuck at. Interest in deep feed forward networks was revived around 2006 by a group of researchers brought together by the Canadian Institute for Advanced Research (CIFAR). The researchers introduced unsupervised learning procedures that could create layers of feature detectors without requiring labelled data. The objective in learning each layer of feature detectors was to be able to reconstruct or model the activities of feature detectors (or raw inputs) in the layer below. By 'pre-training' several layers of progressively more complex feature detectors using this reconstruction objective, the weights of a deep network could be initialized to sensible values. A final layer of output units could then be added to the top of the network and the whole deep system could be fine-tuned using standard backpropagation. This worked remarkably well for recognizing handwritten digits or for detecting pedestrians, especially when the amount of labelled data was very limited.

The first major application of this pre-training approach was in speech recognition, and it was made possible by the advent of fast graphics processing units (GPUs) that were convenient to program and allowed researchers to train networks 10 or 20 times faster. In 2009, the approach was used to map short temporal windows of coefficients extracted from a sound wave to a set of probabilities for the various fragments of speech that might be represented by the frame in the centre of the window.

It achieved record-breaking results on a standard speech recognition benchmark that used a small vocabulary38 and was quickly developed to give record-breaking results on a large vocabulary task39. By 2012, versions of the deep net from 2009 were being developed by many of the major speech groups6 and were already being deployed in Android phones. For smaller data sets, unsupervised pre-training helps to prevent overfitting, leading to significantly better generalization when the number of labelled examples is small, or in a transfer setting where we have lots of examples for some 'source' tasks but very few for some 'target' tasks. Once deep learning had been rehabilitated, it turned out that the pre-training stage was only needed for small data sets. There was, however, one particular type of deep, feedforward network that was much easier to train and generalized much better than networks

with full connectivity between adjacent layers. This was the convolutional neural network (ConvNet). It achieved many practical successes during the period when neural networks were out of favour and it has recently been widely adopted by the computer vision community.

Convolutional neural networks:

ConvNets are designed to process data that come in the form of multiple arrays, for example a colour image composed of three 2D arrays containing pixel intensities in the three colour channels. Many data modalities are in the form of multiple arrays: 1D for signals and sequences, including language; 2D for images or audio spectrograms; and 3D for video or volumetric images. There are four key ideas behind ConvNets that take advantage of the properties of natural signals: local connections, shared weights, pooling and the use of many layers. The architecture of a typical ConvNet is structured as a series of stages. The first few stages are composed of two types of layers: convolutional layers and pooling layers. Units in a convolutional layer are organized in feature maps, within which each unit is connected to local patches in the feature maps of the previous layer through a set of weights called a filter bank. The result of this local weighted sum is then passed through a non-linearity such as a ReLU.

All units in a feature map share the same filter bank. Different feature maps in a layer use different filter banks. The reason for this architecture is twofold. First, in array data such as images, local groups of values are often highly correlated, forming distinctive local motifs that are easily detected. Second, the local statistics of images and other signals are invariant to location. In other words, if a motif can appear in one part of the image, it could appear anywhere, hence the idea of units at different locations sharing the same weights and detecting the same pattern in different parts of the array. Mathematically, the filtering operation performed by a feature map is a discrete convolution, hence the name. Although the role of the convolutional layer is to detect local conjunctions of features from the previous layer, the role of the pooling layer is to merge semantically similar features into one. Because the relative positions of the features forming a motif can vary somewhat, reliably detecting the motif can be done by coarse-graining the position of each feature. A typical pooling unit computes the maximum of a local patch of units in one feature map (or in a few feature maps). Neighbouring

pooling units take input from patches that are shifted by more than one row or column, thereby reducing the dimension of the representation and creating an invariance to small shifts and distortions. Two or three stages of convolution, non-linearity and pooling are stacked, followed by more convolutional and fully-connected layers. Backpropagating gradients through a ConvNet is as simple as through a regular deep network, allowing all the weights in all the filter banks to be trained.

Deep neural networks exploit the property that many natural signals are compositional hierarchies, in which higher-level features are obtained by composing lower-level ones. In images, local combinations of edges form motifs, motifs assemble into parts, and parts form objects. Similar hierarchies exist in speech and text from sounds to phones, phonemes, syllables, words and sentences. The pooling allows representations to vary very little when elements in the previous layer vary in position and appearance. The convolutional and pooling layers in ConvNets are directly inspired by the classic notions of simple cells and complex cells in visual neuroscience, and the overall architecture is reminiscent of the LGN–V1–V2–V4–IT hierarchy in the visual cortex ventral pathway.

When ConvNet models and monkeys are shown the same picture, the activations of high-level units in the ConvNet explains half of the variance of random sets of 160 neurons in the monkey's inferotemporal cortex. ConvNets have their roots in the neocognitron46, the architecture of which was somewhat similar, but did not have an end-to-end supervised-learning algorithm such as backpropagation. A primitive 1D ConvNet called a time-delay neural net was used for the recognition of phonemes and simple words. There have been numerous applications of convolutional networks going back to the early 1990s, starting with time-delay neural networks for speech recognition and document reading. The document reading system used a ConvNet trained jointly with a probabilistic model that implemented language constraints.

By the late 1990s this system was reading over 10% of all the cheques in the United States. A number of ConvNet-based optical character recognition and handwriting recognition systems were later deployed by Microsoft49. ConvNets were also experimented with in the early 1990s for object detection in natural images, including faces and hands, and for face recognition.

Image understanding with deep convolutional networks:

Since the early 2000s, ConvNets have been applied with great success to the detection, segmentation and recognition of objects and regions in images. These were all tasks in which labelled data was relatively abundant, such as traffic sign recognition, the segmentation of biological images particularly for connect omics, and the detection of faces, text, pedestrians and human bodies in natural images. A major recent practical success of ConvNets is face recognition. Importantly, images can be labelled at the pixel level, which will have applications in technology, including autonomous mobile robots and self-driving cars. Companies such as Mobileye and NVIDIA are using such ConvNet-based methods in their upcoming vision systems for cars.

Other applications gaining importance involve natural language understanding and speech recognition. Despite these successes, ConvNets were largely forsaken by the mainstream computer-vision and machine-learning communities until the ImageNet competition in 2012. When deep convolutional networks were applied to a data set of about a million images from the web that contained 1,000 different classes, they achieved spectacular results, almost halving the error rates of the best competing approaches. This success came from the efficient use of GPUs, ReLUs, a new regularization technique called dropout, and techniques to generate more training examples by deforming the existing ones. This success has brought about a revolution in computer vision; ConvNets are now the dominant approach for almost all recognition and detection tasks and approach human performance on some tasks.

A recent stunning demonstration combines ConvNets and recurrent net modules for the generation of image captions. Recent ConvNet architectures have 10 to 20 layers of ReLUs, hundreds of millions of weights, and billions of connections between units. Whereas training such large networks could have taken weeks only two years ago, progress in hardware, software and algorithm parallelization have reduced training times to a few hours.

The performance of ConvNet-based vision systems has caused most major technology companies, including Google, Facebook, Microsoft, IBM, Yahoo!, Twitter and Adobe, as well as a quickly growing number of start-ups to initiate research and development projects and to

deploy ConvNet-based image understanding products and services. ConvNets are easily amenable to efficient hardware implementations in chips or field-programmable gate arrays. A number of companies such as NVIDIA, Mobileye, Intel, Qualcomm and Samsung are developing ConvNet chips to enable real-time vision applications in smartphones, cameras, robots and self-driving cars.

Distributed representations and language processing Deep-learning theory shows that deep nets have two different exponential advantages over classic learning algorithms that do not use distributed representations. Both of these advantages arise from the power of composition and depend on the underlying data-generating distribution having an appropriate componential structure. First, learning distributed representations enable generalization to new combinations of the values of learned features beyond those seen during training (for example, 2n combinations are possible with n binary features).

Second, composing layers of representation in a deep net brings the potential for another exponential advantage (exponential in the depth). The hidden layers of a multilayer neural network learn to represent the network's inputs in a way that makes it easy to predict the target outputs. This is nicely demonstrated by training a multilayer neural network to predict the next word in a sequence from a local context of earlier words. Each word in the context is presented to the network as a one-of-N vector, that is, one component has a value of 1 and the rest are 0. In the first layer, each word creates a different pattern of activations, or word vectors.

In a language model, the other layers of the network learn to convert the input word vectors into an output word vector for the predicted next word, which can be used to predict the probability for any word in the vocabulary to appear as the next word. The network learns word vectors that contain many active components each of which can be interpreted as a separate feature of the word, as was first demonstrated in the context of learning distributed representations for symbols. These semantic features were not explicitly present in the input. They were discovered by the learning procedure as a good way of factorizing the structured relationships between the input and output symbols into multiple 'micro-rules'.

Learning word vectors turned out to also work very well when the word sequences come from a large corpus of real text and the individual micro-rules are unreliable. When trained to predict the next word in a news story, for example, the learned word vectors for Tuesday and

Wednesday are very similar, as are the word vectors for Sweden and Norway. Such representations are called distributed representations because their elements (the features) are not mutually exclusive and their many configurations correspond to the variations seen in the observed data. These word vectors are composed of learned features that were not determined ahead of time by experts, but automatically discovered by the neural network.

Vector representations of words learned from text are now very widely used in natural language applications. The issue of representation lies at the heart of the debate between the logic-inspired and the neural-network-inspired paradigms for cognition. In the logic-inspired paradigm, an instance of a symbol is something for which the only property is that it is either identical or non-identical to other symbol instances. It has no internal structure that is relevant to its use; and to reason with symbols, they must be bound to the variables in judiciously chosen rules of inference.

By contrast, neural networks just use big activity vectors, big weight matrices and scalar non-linearities to perform the type of fast 'intuitive' inference that underpins effortless common-sense reasoning. Before the introduction of neural language models, the standard approach to statistical modelling of language did not exploit distributed representations: it was based on counting frequencies of occurrences of short symbol sequences of length up to N (called N-grams). The number of possible N-grams is on the order of V^N, where V is the vocabulary size, so taking into account a context of more than a handful of words would require very large training corpora. N-grams treat each word as an atomic unit, so they cannot generalize across semantically related sequences of words, whereas neural language models can because they associate each word with a vector of real valued features, and semantically related words end up close to each other in that vector space.

Recurrent neural networks:

When backpropagation was first introduced, its most exciting use was for training recurrent neural networks (RNNs). For tasks that involve sequential inputs, such as speech and language, it is often better to use RNNs. RNNs process an input sequence one element at a time, maintaining in their hidden units a 'state vector' that implicitly contains information about the history of all the past elements of the sequence. When we consider the outputs of

the hidden units at different discrete time steps as if they were the outputs of different neurons in a deep multilayer network, it becomes clear how we can apply backpropagation to train RNNs. RNNs are very powerful dynamic systems, but training them has proved to be problematic because the backpropagated gradients either grow or shrink at each time step, so over many time steps they typically explode or vanish. Thanks to advances in their architecture and ways of training them, RNNs have been found to be very good at predicting the next character in the text83 or the next word in a sequence, but they can also be used for more complex tasks. For example, after reading an English sentence one word at a time, an English 'encoder' network can be trained so that the final state vector of its hidden units is a good representation of the thought expressed by the sentence. This thought vector can then be used as the initial hidden state of (or as extra input to) a jointly trained French 'decoder' network, which outputs a probability distribution for the first word of the French translation.

If a particular first word is chosen from this distribution and provided as input to the decoder network it will then output a probability distribution for the second word of the translation and so on until a full stop is chosen. Overall, this process generates sequences of French words according to a probability distribution that depends on the English sentence. This rather naive way of performing machine translation has quickly become competitive with the state-of-the-art, and this raises serious doubts about whether understanding a sentence requires anything like the internal symbolic expressions that are manipulated by using inference rules. It is more compatible with the view that everyday reasoning involves many simultaneous analogies that each contribute plausibility to a conclusion.

Instead of translating the meaning of a French sentence into an English sentence, one can learn to 'translate' the meaning of an image into an English sentence. The encoder here is a deep ConvNet that converts the pixels into an activity vector in its last hidden layer. The decoder is an RNN similar to the ones used for machine translation and neural language modelling. There has been a surge of interest in such systems recently. RNNs, once unfolded in time (Fig. 5), can be seen as very deep feedforward networks in which all the layers share the same weights. Although their main purpose is to learn long-term dependencies, theoretical and empirical evidence shows that it is difficult to learn to store information for very long. To correct for that, one idea is to augment the network with an explicit memory. The first proposal of this

kind is the long short-term memory (LSTM) networks that use special hidden units, the natural behaviour of which is to remember inputs for a long time. A special unit called the memory cell acts like an accumulator or a gated leaky neuron: it has a connection to itself at the next time step that has a weight of one, so it copies its own real-valued state and accumulates the external signal, but this self-connection is multiplicatively gated by another unit that learns to decide when to clear the content of the memory. LSTM networks have subsequently proved to be more effective than conventional RNNs, especially when they have several layers for each time step, enabling an entire speech recognition system that goes all the way from acoustics to the sequence of characters in the transcription. LSTM networks or related forms of gated units are also currently used for the encoder and decoder networks that perform so well at machine translation.

Over the past year, several authors have made different proposals to augment RNNs with a memory module.

Proposals include the Neural Turing Machine in which the network is augmented by a 'tape-like' memory that the RNN can choose to read from or write to88, and memory networks, in which a regular network is augmented by a kind of associative memory89. Memory networks have yielded excellent performance on standard question-answering benchmarks. The memory is used to remember the story about which the network is later asked to answer questions. Beyond simple memorization, neural Turing machines and memory networks are being used for tasks that would normally require reasoning and symbol manipulation. Neural Turing machines can be taught algorithms.

Among other things, they can learn to output a sorted list of symbols when their input consists of an unsorted sequence in which each symbol is accompanied by a real value that indicates its priority in the list.

Memory networks can be trained to keep track of the state of the world in a setting similar to a text adventure game and after reading a story, they can answer questions that require complex inference.

In one test example, the network is shown a 15-sentence version of the Lord of the Rings and correctly answers questions such as "where is Frodo now?".

The future of deep learning:

Unsupervised learning had a catalytic effect in reviving interest in deep learning, but has since been overshadowed by the successes of purely supervised learning. Although we have not focused on it in this Review, we expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object. Human vision is an active process that sequentially samples the optic array in an intelligent, task-specific way using a small, high-resolution fovea with a large, low-resolution surround. We expect much of the future progress in vision to come from systems that are trained end-to-end and combine ConvNets with RNNs that use reinforcement learning to decide where to look. Systems combining deep learning and reinforcement learning are in their infancy, but they already outperform passive vision systems at classification tasks and produce impressive results in learning to play many different video games.

Natural language understanding is another area in which deep learning is poised to make a large impact over the next few years. We expect systems that use RNNs to understand sentences or whole documents will become much better when they learn strategies for selectively attending to one part at a time. Ultimately, major progress in artificial intelligence will come about through systems that combine representation learning with complex reasoning. Although deep learning and simple reasoning have been used for speech and handwriting recognition for a long time, new paradigms are needed to replace rule-based manipulation of symbolic expressions by operations on large vectors

1.2 SCOPE OF THE PROJECT:

This paper aims at investigating the principles, methodologies and algorithms for detecting fake news articles, creators and subjects from online social networks and evaluating the corresponding performance.

1.3 OBJECTIVE:

This paper addresses the challenges introduced by the unknown characteristics of fake news and diverse connections among news articles, creators and subjects. This paper introduces a novel gated graph neural network, namely FAKEDETECTOR. Based on a set of explicit and latent features extracted from the textual information, FAKEDETECTOR builds a deep diffusive network model to learn the representations of news articles, creators and subjects simultaneously. Extensive experiments have been done on a real-world fake news dataset to compare FAKEDETECTOR with several state-of-the-art models, and the experimental results are provided in the full-version of this paper.

1.4 PROBLEM STATEMENT:

In this paper, we will study the fake news detection (including articles, creators and subjects) problem in online social networks. Based on various types of heterogeneous information sources, including both textual contents/ profile/descriptions and the authorship and article-subject relationships among them, we aim at identifying fake news from the online social networks simultaneously. We formulate the fake news detection problem as a credibility inference problem, where the real ones will have a higher credibility while unauthentic ones will have a lower one instead.

To solve the problem, in this paper, we will introduce a new graph neural network model, namely FAKEDETECTOR. In FAKEDETECTOR, the fake news detection problem is formulated as a credibility label inference problem, and FAKEDETECTOR aims at learning a prediction model to infer the credibility labels of news articles, creators and subjects simultaneously. FAKEDETECTOR deploys a new hybrid feature learning unit (HFLU) for learning the explicit and latent feature representations of news articles, creators and subjects respectively, and introduce a novel deep diffusive network model with the gated diffusive unit for the heterogeneous information fusion within the social networks

1.5 EXISTING SYSTEM:

- In December of 2016, a group of volunteers from industry and academia started a contest called the Fake News Challenge. The goal of this contest was to encourage the development of tools that may help human fact checkers identify deliberate misinformation in news stories through the use of machine learning, natural language processing and artificial intelligence. The organizers decided that the first step in this overarching goal was understanding what other news organizations are saying about the topic in question. As such, they decided that stage one of their contest would be a stance detection competition.
- More specifically, the organizers built a dataset of headlines and bodies of text and challenged competitors to build classifiers that could correctly label the stance of a body text, relative to a given headline, into one of four categories: "agree", "disagree", "discusses" or "unrelated." The top three teams all reached over 80% accuracy on the test set for this task. The top team's model was based on a weighted average between gradient-boosted decision trees and a deep convolutional neural network.

1.5.1 EXISTING SYSTEM DISADVANTAGES

- **Impact on society:** spams usually exist in personal emails or specific review websites and merely have a local impact on a small number of audiences, while the impact fake news in online social networks can be tremendous due to the massive user numbers globally, which is further boosted by the extensive information sharing and propagation among these users.
- **Audiences' initiative:** instead of receiving spam emails passively, users in online social networks may seek for, receive and share news information actively with no sense about its correctness.
- **Identification difficulty:** via comparisons with abundant regular messages (in emails or review websites), spams are usually easier to be distinguished, whereas identifying fake news with erroneous information is incredibly challenging, since it requires both tedious

evidence-collecting and careful fact-checking due to the lack of other comparative news articles available.

1.5.2 LITERATURE SURVEY

Title: Opinion fraud detection in online reviews by network effects

Author: L. Akoglu, R. Chandy, and C. Faloutsos

Year: 2013

Description:

User-generated online reviews can play a significant role in the success of retail products, hotels, restaurants, etc. However, review systems are often targeted by opinion spammers who seek to distort the perceived quality of a product by creating fraudulent reviews. We propose a fast and effective framework, FRAUDEAGLE, for spotting fraudsters and fake reviews in online review datasets. Our method has several advantages:

(1) it exploits the network effect among reviewers and products, unlike the vast majority of existing methods that focus on review text or behavioral analysis,

(2) it consists of two complementary steps; scoring users and reviews for fraud detection, and grouping for visualization and sensemaking,

(3) it operates in a completely unsupervised fashion requiring no labeled data, while still incorporating side information if available, and

(4) it is scalable to large datasets as its run time grows linearly with network size. We demonstrate the effectiveness of our framework on synthetic and real datasets; where FRAUDEAGLE successfully reveals fraud-bots in a large online app review database.

Title: Social media and fake news in the 2016 election.

Author: H. Allcott and M. Gentzkow

Year: 2017

Description:

Following the 2016 U.S. presidential election, many have expressed concern about the effects of false stories ("fake news"), circulated largely through social media. We discuss the economics of fake news and present new data on its consumption prior to the election. Drawing on web browsing data, archives of fact-checking websites, and results from a new online survey, we find:

(i) social media was an important but not dominant source of election news, with 14 percent of Americans calling social media their "most important" source;

(ii) of the known false news stories that appeared in the three months before the election, those favouring Trump were shared a total of 30 million times on Facebook, while those favouring Clinton were shared 8 million times;

(iii) the average American adult saw on the order of one or perhaps several fake news stories in the months around the election, with just over half of those who recalled seeing them believing them; and

(iv) people are much more likely to believe stories that favor their preferred candidate, especially if they have ideologically segregated social media networks.

Title: Empirical evaluation of gated recurrent neural networks on sequence modelling

Author: J. Chung, C¸ G¨ulc¸ehre, K. Cho, and Y. Bengio

Year: 2014

Description:

In this paper we compare different types of recurrent units in recurrent neural networks (RNNs). Especially, we focus on more sophisticated units that implement a gating mechanism, such as a long short-term memory (LSTM) unit and a recently proposed gated recurrent unit (GRU). We evaluate these recurrent units on the tasks of polyphonic music modelling and speech signal modelling. Our experiments revealed that these advanced recurrent units are indeed better than more traditional recurrent units such as tanh units. Also, we found GRU to be comparable to LSTM.

Title: Detecting and characterizing social spam campaigns

Author: H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Zhao

Year: 2010

Description:

Online social networks (OSNs) are popular collaboration and communication tools for millions of users and their friends. Unfortunately, in the wrong hands, they are also effective tools for executing spam campaigns and spreading malware. Intuitively, a user is more likely to respond to a message from a Facebook friend than from a stranger, thus making social spam a more effective distribution mechanism than traditional email. In fact, existing evidence shows malicious entities are already attempting to compromise OSN account credentials to support these "high-return" spam campaigns.

In this paper, we present an initial study to quantify and characterize spam campaigns launched using accounts on online social networks. We study a large anonymized dataset of asynchronous "wall" messages between Facebook users. We analyze all wall messages received by roughly 3.5 million Facebook users (more than 187 million messages in all), and use a set of automated techniques to detect and characterize coordinated spam campaigns. Our system detected roughly 200,000 malicious wall posts with embedded URLs, originating from more than 57,000 user accounts. We find that more than 70% of all malicious wall posts advertise phishing sites. We also study the characteristics of malicious accounts, and see that more than 97% are compromised accounts, rather than "fake" accounts created solely for the purpose of spamming. Finally, we observe that, when adjusted to the local time of the sender, spamming dominates actual wall post activity in the early morning hours, when normal users are asleep.

Title: Convolutional neural networks for sentence classification

Author: Y. Kim

Year: 2014

Description:

We report on a series of experiments with convolutional neural networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks. We show that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks. Learning task-specific vectors through fine-tuning offers further gains in performance. We additionally propose a simple modification to the architecture to allow for the use of both task-specific and static vectors. The CNN models discussed herein improve upon the state of the art on 4 out of 7 tasks, which include sentiment analysis and question classification.

1.6 PROPOSED SYSTEM

- In this paper, we will study the fake news detection (including articles, creators and subjects) problem in online social networks. Based on various types of heterogeneous information sources, including both textual contents/ profile/descriptions and the authorship and article-subject relationships among them, we aim at identifying fake news from the online social networks simultaneously.
- We formulate the fake news detection problem as a credibility inference problem, where the real ones will have a higher credibility while unauthentic ones will have a lower one instead.
- To solve the problem, in this paper, we will introduce a model, namely FAKEDETECTOR. In FAKEDETECTOR, the fake news detection problem is formulated as a credibility label inference problem, and FAKEDETECTOR aims at learning a prediction model to infer the credibility labels of news articles, creators and subjects simultaneously. FAKEDETECTOR deploys a new hybrid feature learning unit (HFLU) for learning the explicit and latent feature representations of news articles, creators and subjects respectively.

1.6.1 PROPOSED SYSTEM ADVANTAGES:

- The main contribution of this project is support for the idea that machine learning could be useful in a novel way for the task of classifying fake news.
- Furthermore, the model seems to be relatively unphased by the exclusion of certain "giveaway" topic words in the training set, as it is able to pick up on trigrams that are less specific to a given topic, if need be. As such, this seems to be a really good start on a tool that would be useful to augment humans ability to detect Fake News

CHAPTER 2
PROJECT DESCRIPTION

2.1 GENERAL

We will provide the detailed information about the FAKEDETECTOR model in this section. FAKEDETECTOR covers two main components: representation feature learning, and credibility label inference, which together will compose the deep diffusive network model FAKEDETECTOR.

2.2 METHODOLOGIES

2.2.1 MODULES NAME:

This project having the following five modules:

- Data Collection
- Dataset
- Data Preparation
- Model Selection
- Analyze and Prediction
- Accuracy on test set
- Saving the Trained Model

2.2.2 MODULES EXPLANATION AND DIAGRAM

- **Data Collection:**

This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get, the better our model will perform.

There are several techniques to collect the data, like web scraping, manual interventions and etc. The dataset used in this Fake-news Detection taken from Kaggle

Link: https://www.kaggle.com/c/fake-news/data

➢ Dataset:

The dataset consists of 20800 individual data. There are 5 columns in the dataset, which are described below

1. Id: unique id for a news article
2. Title: the title of a news article
3. Author: author of the news article
4. Text: the text of the article; could be incomplete
5. Label: a label that marks the article as potentially unreliable
   1: unreliable
   0: reliable

➢ Data Preparation:

We will transform the data. By getting rid of missing data and removing some columns. First, we will create a list of column names that we want to keep or retain. Next, we drop or remove all columns except for the columns that we want to retain. Finally, we drop or remove the rows that have missing values from the data set.

Steps to follow:

1. Removing extra symbols
2. Removing punctuations
3. Removing the stop words
4. Stemming
5. Tokenization
6. Feature extractions
7. TF-IDF vectorizer
8. Counter vectorizer with TF-IDF transformer

➢ Model Selection:

We used Passive-Aggressive algorithm but in base paper neural network algorithm is given, The Passive-Aggressive algorithms are a family of Machine learning algorithms that are not very well known by beginners and even intermediate Machine Learning enthusiasts. However, they can be very useful and efficient for certain applications so we applied.

1. How Passive-Aggressive Algorithms Work:

Passive-Aggressive algorithms are called so because:

Passive: If the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model.

Aggressive: If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.

Understanding the mathematics behind this algorithm is not very simple and is beyond the scope of a single article. This article provides just an overview of the algorithm and a simple implementation of it. To learn more about the mathematics behind this algorithm.

2. Important parameters:

C: This is the regularization parameter, and denotes the penalization the model will make on an incorrect prediction

max_iter: The maximum number of iterations the model makes over the training data.

tol: The stopping criterion. If it is set to None, the model will stop when (*loss* > *previous_loss* – *tol*). By default, it is set to 1e-3

➢ Analyze and Prediction:

In the actual dataset, we chose only 2 features:

1. Text: the text of the article; could be incomplete
2 Label: a label that marks the article as potentially unreliable
    1: FAKE
    0: REAL

➢ Accuracy on test set:

We got an accuracy of 70.2% on test set.

➢ Saving the Trained Model:

Once you're confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or. pkl file using a library like pickle.
Make sure you have pickle installed in your environment.
Next, let's import the module and dump the model into. pkl file

2.2.3 GIVEN INPUT EXPECTED OUTPUT:

➢ INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

➢ OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

- **OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be dive.

loped while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER 3
REQUIREMENTS ENGINEERING

3.1 GENERAL

These are the requirements for doing the project. Without using these tools and software's we can't do the project. So, we have two requirements to do the project. They are

1. Hardware Requirements.
2. Software Requirements.

3.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system does and not how it should be implemented.

HARDWARE

- Processor : Pentium Iv 2.6 Ghz, Intel Core 2 Duo.
- Ram : 512 Mb Dd Ram
- Monitor : 15" Color
- Hard Disk : 40 GB

3.3 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

SOFTWARE

- Front End : J2EE (JSP, Servlet)
- Back End : MY SQL 5.5
- Operating System : Windows 7
- IDE : Eclipse

3.4 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behaviour, and outputs. This paper aims at investigating the principles, methodologies and algorithms for detecting fake news articles, creators and subjects from online social networks and evaluating the corresponding performance. This paper addresses the challenges introduced by the unknown characteristics of fake news and diverse connections among news articles, creators and subjects

3.5 NON-FUNCTIONAL REQUIREMENTS

The major non-functional Requirements of the system are as follows

- **Usability**

The system is designed with completely automated process hence there is no or less user intervention.

- **Reliability**

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.

- **Performance**

This system is developing in the high level languages and using the advanced front-end and back-end technologies it will give response to the end user on client system with in very less time.

- **Supportability**

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

- **Implementation**

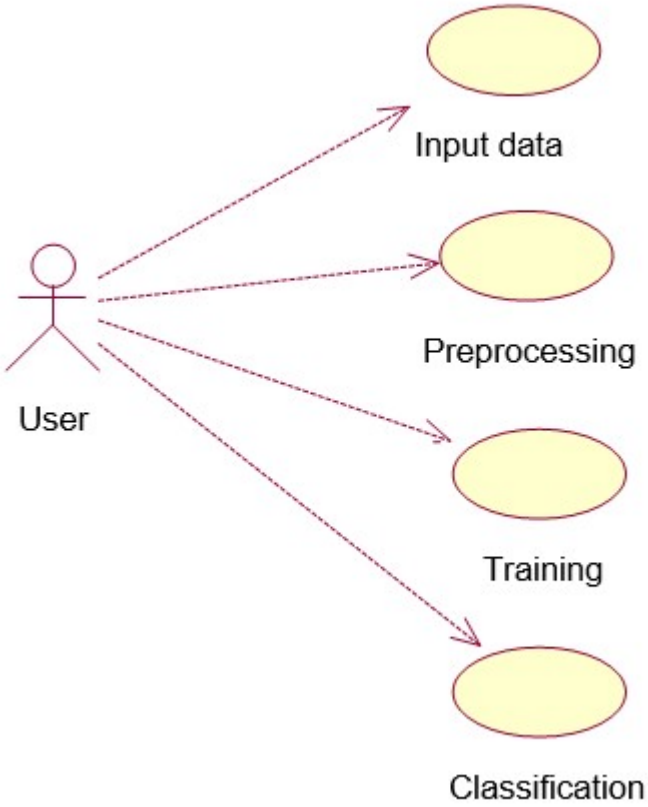
The system is implemented in web environment using struts framework. The apache tomcat is used as the web server and windows xp professional is used as the platform. Interface the user interface is based on Struts provides HTML Tag

CHAPTER 4
DESIGN ENGINEERING

4.1 GENERAL

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

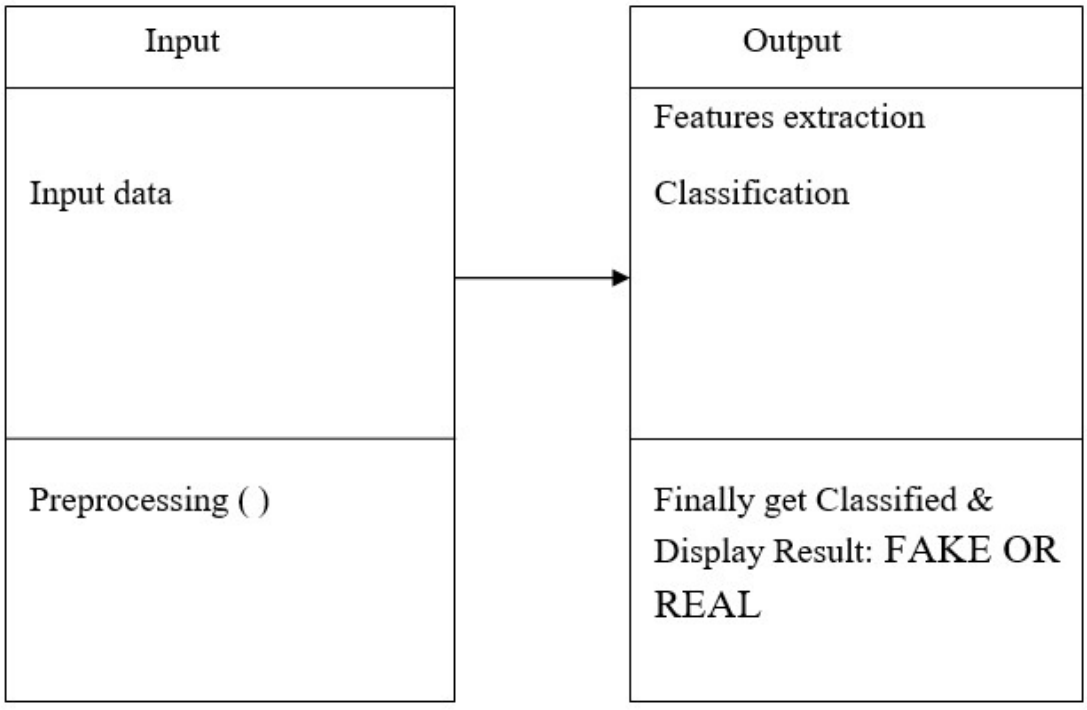
4.1.1 Use Case Diagram



EXPLANATION:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral 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. Roles of the actors in the system can be depicted.

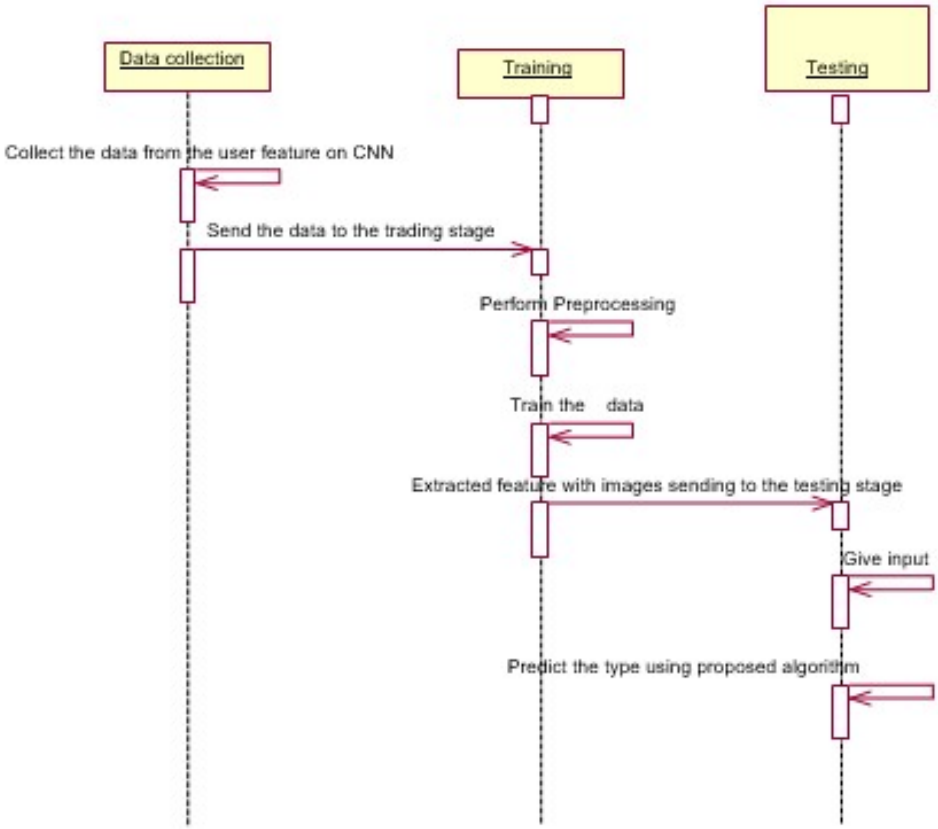
4.1.2 Class Diagram



EXPLANATION

In software engineering, a class diagram in the Unified Modelling 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.

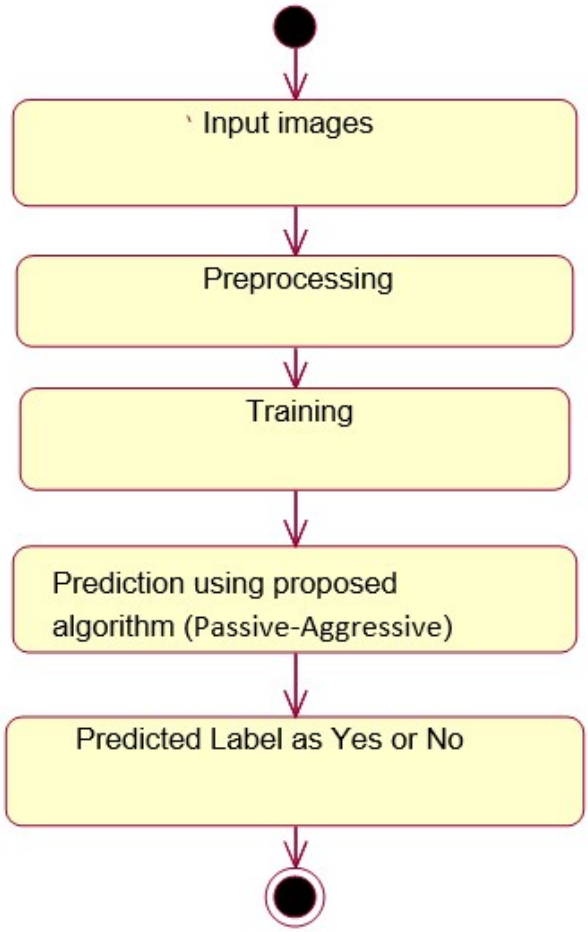
4.1.3 Sequence Diagram



EXPLANATION:

A sequence diagram in Unified modelling 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.

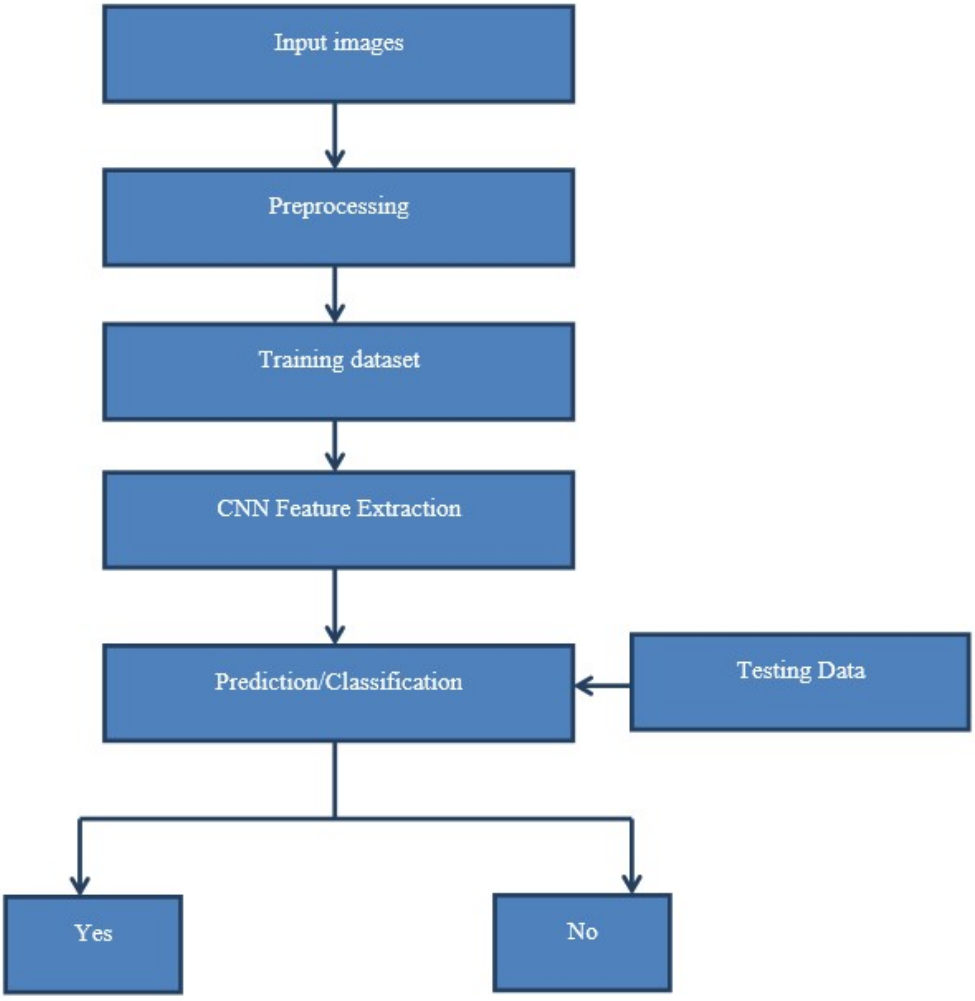
4.1.4 Activity Diagram



EXPLANATION:

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.

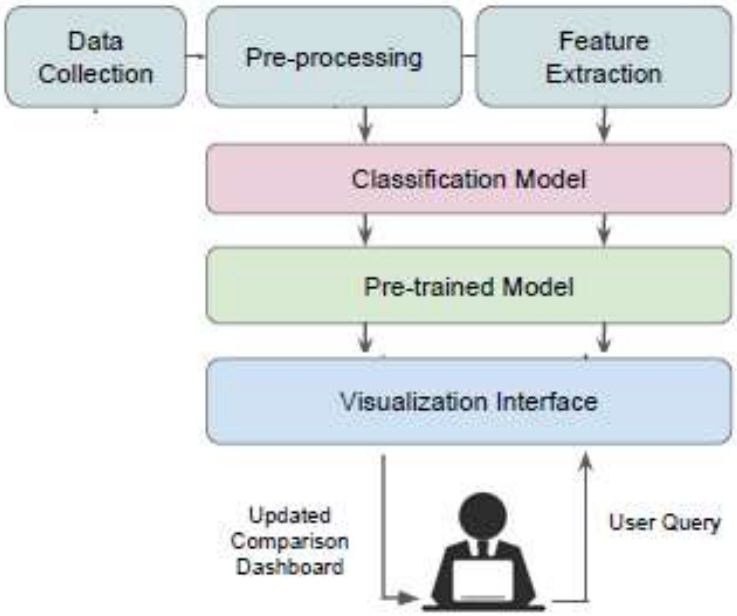
4.1.5 Data Flow Diagram:



EXPLANATION:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

4.2 System Architecture



CHAPTER 5
DEVELOPMENT TOOLS

5.1 GENERAL

Python:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link for the Windows installer python-XYZ.msifile where XYZ is the version you need to install.
- To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

First Python Program

Let us execute programs in different modes of programming.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

```
$ python

Python2.4.3(#1,Nov112010,13:34:43)

[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2

Type"help","copyright","credits"or"license"for more information.
```

Type the following text at the Python prompt and press the Enter −

```
>>print"Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**. However in Python version 2.4.3, this produces the following result −

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file −

```
print"Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

This produces the following result −

```
Hello, Python!
```

Flask Framework:

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods −

Sr.No	Methods & Description
1	**GET** Sends data in unencrypted form to the server. Most common method.
2	**HEAD** Same as GET, but without response body

3	**POST** Used to send HTML form data to server. Data received by POST method is not cached by server.
4	**PUT** Replaces all current representations of the target resource with the uploaded content.
5	**DELETE** Removes all current representations of the target resource given by a URL

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

```
<html>

<body>

<formaction="http://localhost:5000/login"method="post">

<p>Enter Name:</p>

<p><inputtype="text"name="nm"/></p>

<p><inputtype="submit"value="submit"/></p>
```

```
</form>

</body>

</html>
```

Now enter the following script in Python shell.

```
from flask importFlask, redirect,url_for, request

app=Flask(__name__)

@app.route('/success/<name>')

def success(name):

return'welcome %s'% name

@app.route('/login',methods=['POST','GET'])

def login():

ifrequest.method=='POST':

user=request.form['nm']

return redirect(url_for('success',name= user))

else:

user=request.args.get('nm')

return redirect(url_for('success',name= user))

if __name__ =='__main__':

app.run(debug =True)
```

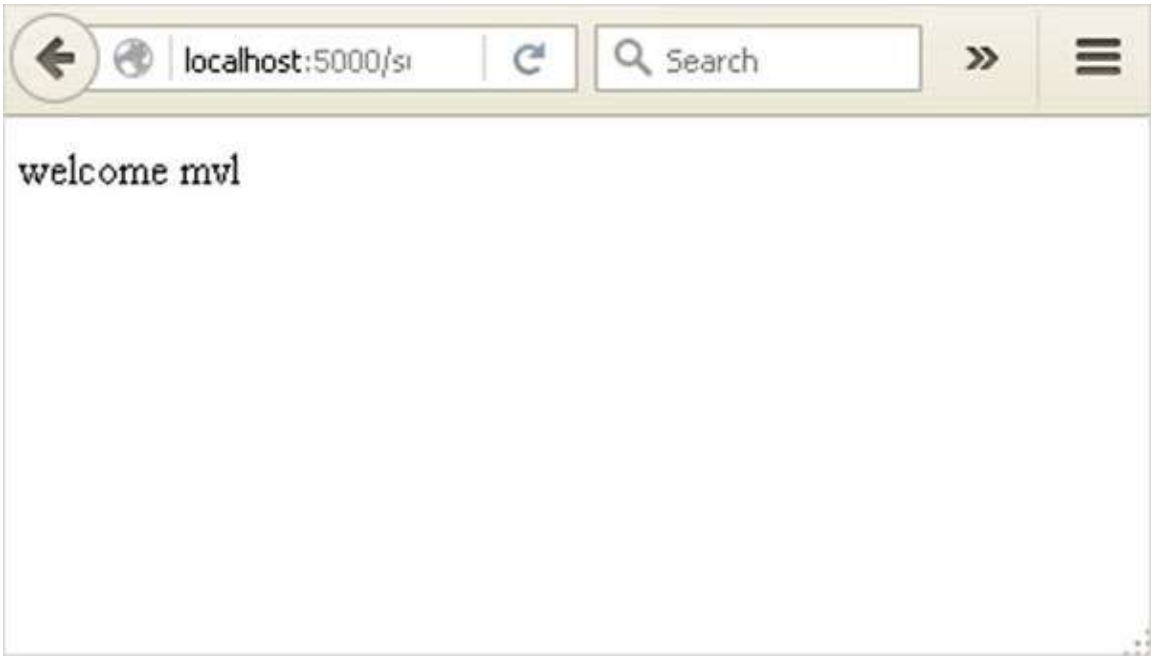
After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.

Form data is POSTed to the URL in action clause of form tag.

http://localhost/login is mapped to the **login()** function. Since the server has received data by **POST** method, value of 'nm' parameter obtained from the form data is obtained by −

```
user = request.form['nm']
```

It is passed to **'/success'** URL as variable part. The browser displays a **welcome** message in the window.



Change the method parameter to **'GET'** in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of 'nm' parameter is now obtained by −

```
User = request.args.get('nm')
```

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to 'nm' parameter is passed on to '/success' URL as before.

What is Python?

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https://www.python.org/

Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

```
helloworld.py
```

```
print("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
```

Which will write "Hello, World!" in the command line:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type"help", "copyright", "credits" or "license" for more information.
```

```
>>>print("Hello,World!")
Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

```
exit()
```

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

>>>print("Hello,World!")

Hello, World!

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

C:\Users*Your Name*>python myfile.py

Python Indentations

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.

Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
  print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

```
if 5 > 2:
print("Five is greater than two!")
```

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

Example

Comments in Python:

```
#This is a comment.
print("Hello, World!")
```

Docstrings

Python also has extended documentation capability, called docstrings.

Docstrings can be one line, or multiline.

Python uses triple quotes at the beginning and end of the docstring:

Example

Docstrings are also comments:

```
"""This is a
multiline docstring."""
print("Hello, World!")
```

CHAPTER 6
IMPLEMENTATION

6.1 GENERAL

The Implementation is nothing but sores code of project.

Structure

Fake detector

- Model
  - fake_news_detector_prepare_model.ipynb
- Static
- Templates
  - Chart.HTML
  - First.HTML
  - Future .HTML
  - Index.HTML
  - Login.HTML
  - Preview.HTML
  - Upload.HTML
- App.py
- Docker file
- Train.css
- Upload.css

6.2 IMPLEMENTATION

Coding:

App.py

```
from flask import Flask, render_template, request
import pickle
import pandas as pd

# create Flask application
app = Flask(__name__)

# read object TfidfVectorizer and model from disk
vec = pickle.load(open('vectorizer.pkl', 'rb'))
model = pickle.load(open('model.pkl', 'rb'))
```

```
@app.route('/')
@app.route('/first')
def first():
    return render_template('first.html')
@app.route('/login')
def login():
    return render_template('login.html')
@app.route('/chart')
def chart():
    return render_template('chart.html')
@app.route('/abstract')
def abstract():
    return render_template('first.html')
@app.route('/future')
def future():
    return render_template('future.html')
@app.route('/upload')
def upload():
    return render_template('upload.html')
@app.route('/preview',methods=["POST"])
def preview():
    if request.method == 'POST':
        dataset = request.files['datasetfile']
        df = pd.read_csv(dataset,encoding = 'unicode_escape')
        df.set_index('Id', inplace=True)
        return render_template("preview.html",df_view = df)


@app.route('/home')
def home():
    return render_template('index.html')


@app.route('/predict', methods=['POST'])
def predict():
    error = None
    if request.method == 'POST':
        # message
        msg = request.form['message']
        msg = pd.DataFrame(index=[0], data=msg, columns=['data'])

        # transform data
        text = vec.transform(msg['data'].astype('U'))
```

```
        # model
        result = model.predict(text)

        if result == 0:
            result = "real"
        else:
            result = 'fake'

        return render_template('index.html', prediction_value=result)
    else:
        error = "Invalid message"
        return render_template('index.html', error=error)


if __name__ == "__main__":
    app.run(debug=True)
```

DOCKER FILE

```
FROM python:3

# We copy just the requirements.txt first to leverage Docker cache
COPY ./requirements.txt /app/requirements.txt

WORKDIR /app

RUN pip install --no-cache-dir -r requirements.txt

COPY . /app

CMD ["python", "./app.py"]
```

fake_news_detector_prepare_model.ipynb

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pickle

from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB


from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix

# from sklearn.model_selection import GridSearchCV
# import plotly.express as px
# import plotly.express as px
# import plotly.figure_factory as ff

sns.set()

# Read data

train = pd.read_csv('data/train.csv')
test = pd.read_csv('data/test.csv')
y_test = pd.read_csv('data/submit.csv')

test['label'] = y_test['label']

train.head()

# Delete NaN value
train.info()
test.info()
train.dropna(axis=0, how='any', inplace=True)
test.dropna(axis=0, how='any', inplace=True)
print('Number of NaN values in sets: ', train.isna().sum().sum() + test.isna(
).sum().sum())
map_values = {
    0: 'Real',
    1: 'Fake'
}

df_t = train.label.value_counts()
df_t.index = df_t.index.map(map_values)

plt.figure(figsize=(12, 6))
df_t.plot(kind='bar', fontsize=14)
```

```
plt.title("Real vs fake news", fontsize=20)
plt.show()
```

Create object TfidfVectorizer

```
vectorizer = TfidfVectorizer()

train_vecotorizer = vectorizer.fit_transform(train['text'].astype('U'))
test_vecotorizer = vectorizer.transform(test['text'].astype('U'))
```

Create models

PassiveAggressiveClassifier

```
pac = PassiveAggressiveClassifier(max_iter=88)
pac.fit(train_vecotorizer, train['label'] )

# params_grid = {
#     'max_iter': np.arange(50, 200, 1)
# }

# grid_search = GridSearchCV(pac, params_grid, scoring='accuracy', n_jobs=-
1, cv=5)
# grid_search.fit(train_vecotorizer, train['label'] )



# grid_search.best_params_
pac_score = pac.score(test_vecotorizer, test['label'].values)
pac_score
```

Logistic Regression

```
log_reg = LogisticRegression()

log_reg.fit(train_vecotorizer, train['label'])
lg_score = log_reg.score(test_vecotorizer, test['label'])
lg_score
```

RandomForestClassifier

```
rando= RandomForestClassifier(n_estimators=5)
rando.fit(train_vecotorizer, train['label'])
rando_score = rando.score(test_vecotorizer ,test['label'])
rando_score
```

MultinomialNB

```
classifier = MultinomialNB()
classifier.fit(train_vecotorizer, train['label'] )
multi_score = classifier.score(test_vecotorizer, test['label'])
multi_score
```

Choosing the best model

```
models_value = {'PassiveAggressiveClassifier': pac_score, 'Logistic Regressio
n': lg_score, 'RandomForestClassifier': rando_score, 'MultinomialNB': multi_s
core}
models = pd.Series(models_value).sort_values(ascending=False)
models
```

```
models_list = [log_reg, rando, pac, classifier]
```

```
# px.bar(df_models, x='model', y='value', color='value', width=800, height=50
0, title="Models rank")
plt.figure(figsize=(6, 4))
models.plot(kind='bar')
plt.title('Accuracy of models')
plt.show()
```

```
y_pred_rando= rando.predict(test_vecotorizer)
y_pred_pca= pac.predict(test_vecotorizer)

cm_pac = confusion_matrix(y_pred_rando, test['label'])
cm_rando = confusion_matrix(y_pred_pca, test['label'])
```

```
plot_confusion_matrix(cm_pac)
plt.show()


plot_confusion_matrix(cm_rando)
plt.show()
```

Test models on test data (outer data)

```
fake = pd.read_csv('data/production/Fake.csv')
true = pd.read_csv('data/production/True.csv')
true.head()
fake.head()


def predict_data(data, vectorizer, classifier):
    text = vectorizer.transform(data.astype('U'))
    y_pred = classifier.predict(text)
    return (round(y_pred.sum() / len(data) * 100, 2))


def exec_models(models_name, models):
    df = pd.DataFrame(data=0, columns=['Fake news predict as fake', 'True new
s predict as true'], index=models_name)

    for i in range(len(models)):
        f = predict_data(fake['text'], vectorizer, models[i])
        t = predict_data(true['text'], vectorizer, models[i])

        df.loc[models_name[i]] = [f, t]

        print('\n{}'.format(models_name[i]))
        print("Fake news predict as fake: {}%".format( f  ))
        print("True news predict as true: {}%".format( 100 - t ))

    return df

df_result = exec_models(list(models.index), models_list)
df_result
```

```
df_result['mean'] = df_result[['Fake news predict as fake', 'True news predic
t as true']].mean(axis=1)
df_result = df_result.sort_values(by='mean', ascending=False)


plt.figure(figsize=(6, 4))
_ = df_result['mean'].plot(kind='bar')
plt.show()


# saving models to disk
pickle.dump(vectorizer, open('vectorizer.pkl', 'wb'))
pickle.dump(pac, open('model.pkl', 'wb'))


# loading models from disk
vec = pickle.load( open('vectorizer.pkl', 'rb') )
model = pickle.load( open('model.pkl', 'rb') )
```

Checking models

```
df_models.modelmsg = pd.DataFrame(index=[0], data=true['text'][100], columns=
['data'])
text = vec.transform(msg['data'].astype('U'))
result = model.predict(text)

print("Real (0) Fake (1) news : {}".format( result[0]  ))
```

Fake news

```
msg = pd.DataFrame(index=[0], data=fake['text'][100], columns=['data'])
text = vec.transform(msg['data'].astype('U'))
result = model.predict(text)

print("Real (0) Fake (1) news : {}".format( result[0]  ))
```

Chart.HTML

```
   <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader
.js"></script>
    <script type="text/javascript">
      google.charts.load("current", {packages:["corechart"]});
      google.charts.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Task', 'Hours per Day'],
          ['FAKE',     11],

          ['REAL',    7]
        ]);

        var options = {
          title: '',
          is3D: true,
        };

        var chart = new google.visualization.PieChart(document.getElementById
('piechart_3d'));
        chart.draw(data, options);
      }

    </script>
  </head>
  <body>
    <h1>Fake News Detection (pie chart analysis)</h1>

    <div id="piechart_3d" style="width: 900px; height: 500px; "></div>
  </body>
    <a href="{{url_for('future')}}"><button type="submit" style="background-
color:green;color:white;width:150px;height:40px;">Future</button></a>
</html>
```

First.HTML

```
<html>
    <head>
        <title>Fake News Detection</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-
scale=1" />
        <link rel="stylesheet" href="../static/css/main.css" />
    </head>
    <body>

        <!-- Header -->
            <header id="header" class="alt">
                <div class="logo"><a href="#">Fake  <span>News Detection</spa
n></a></div>
                <a href="#menu" class="toggle"><span>Menu</span></a>
            </header>

        <!-- Nav -->
            <nav id="menu">
                <ul class="links">
                    <li><a href="#">Home</a></li>
                    <li><a href="#">Abstract</a></li>
                    <li><a href="{{ url_for('login') }}">Login</a></li>
                    <li><a href="{{ url_for('upload') }}">Upload data</a></li
>
                    <li><a href="{{ url_for('future') }}">Future</a></li>
                    <li><a href="{{ url_for('chart') }}">chart</a></li>
                    <li><a href="{{ url_for('home') }}">Prediction</a></li>
                </ul>
            </nav>

        <!-- Banner -->
        <!--
            To use a video as your background, set data-
video to the name of your video without
            its extension (eg. images/banner). Your video must be available i
n both .mp4 and .webm
            formats to work correctly.
        -->
            <section id="banner" data-video="../static/images/video2">
                <div class="inner">
```

```
                    <h1>FAKEDETECTOR: Effective Fake News Detection
with Deep Diffusive Neural Network
</h1>

                    <a href="#one" class="button special scrolly">Abstract</a
>
                </div>
            </section>

        <!-- One -->
            <section id="one" class="wrapper style2">
                <div class="inner">
                    <div>
                        <div class="box">

                            <div class="content">
                                <header class="align-center">
                                    <h2>Abstract</h2>

                                </header>
                                <hr />
                                <p> In recent years, due to the booming
Development of online social networks, fake news for various commercial and political purposes has been appearing in large numbers and widespread in the online world. With deceptive words, online social network users can get infected by this online fake news easily, which has brought about tremendous effects on the offline society already.
An important goal in improving the trustworthiness of information in online social networks is to identify the fake news timely. This paper aims at investigating the principles, methodologies and algorithms for detecting fake news articles, creators and subjects from online social networks and evaluating the corresponding performance. This paper addresses the challenges introduced by the unknown characteristics of fake news and diverse connections among news articles, creators
and subjects. This paper introduces a novel gated graph neural network, namely FAKEDETECTOR. Based on a set of explicit and latent features extracted from the textual information, FAKEDETECTOR builds a deep diffusive network model to
learn the representations of news articles, creators and subjects simultaneously. Extensive experiments have been done on a real world fake news dataset to compare FAKEDETECTOR with several state-of-the-art models
</p>

                            </div>
```

```
                    </div>
                </div>
            </div>
        </section>


    <!-- Scripts -->
        <script src="../static/js/jquery.min.js"></script>
        <script src="../static/js/jquery.scrolly.min.js"></script>
        <script src="../static/js/jquery.scrollex.min.js"></script>
        <script src="../static/js/skel.min.js"></script>
        <script src="../static/js/util.js"></script>
        <script src="../static/js/main.js"></script>

    </body>
</html>
```

Future .HTML

```
<html>
<head>
<style>

h1 {
  text-align: center;
  text-transform: uppercase;
  color: red;
}

p {
  text-indent: 50px;
  text-align: justify;
  letter-spacing: 3px;
}
 .button {
  background-color: #4CAF50; /* Green */
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
```

```
  font-size: 16px;
  margin: 4px 2px;
  cursor: pointer;
}

</style>
</head>
<body>

<div>
  <h1>Conclusion and Future Work</h1>
  <p>
```

The introduced GDU model also works for both the news subjects and creator nodes in the network. When applying the GDU to model the states of the subject/creator nodes with two input only, the remaining input port can be assigned with a default value (usually vector 0). Based on the GDU, we can denote the overall architecture of the FAKEDETECTOR as shown in Figure 4, where the lines connecting the GDUs denote the data flow among the unit models. In the following section, we will introduce how to learn the parameters involved in the FAKEDETECTOR model for concurrent credibility inference of multiple nodes.

```
</P>


</div>
 <input type="button" class="button" onclick="window.location.href='{{url_for
('first')}}';" value="HOME" />
</body>
</html>
```

Index.HTML

```
>
<head>
    <meta charset="UTF-8">
    <title>Fake news detector</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.
0.0/css/bootstrap.min.css"
          integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin
="anonymous">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
```

```
    <style>
        span.p_text {
          display: inline-block;
          width: 100px;
          font-size: large;
          font-weight: bold;
          color:brown;
          height: 100px;
          padding: 10px;
          margin-left:10px;
        }

        body  {

background-color: #cccccc;
}
.traffic_title{
margin-left: 40%;
}
    </style>
</head>
<body>
<div class="container">
    <div class="row">
        <div class="col-2"></div>
        <div class="col-8 text-left">
            <h1>Fake news detector</h1>
            <hr>

            <p class="container border-
left">Predict whether the message (english) is real or fake</p>
            <br>
            <br>

            <h5>Enter message: </h5>
            <form action="{{ url_for('predict') }}" method="post">
                <div class="form-group">
                    <textarea id="message" class="form-
control" name="message" rows="4" cols="50" required></textarea><br>
                    <button id='btn1' type="submit" class="btn btn-lg btn-
primary">Predict</button>

                </div>
```

```
            </form>
            <br>
            <br>

            <div id="s" class="alert alert-
success" role="alert" style="display:inline">The news is: </div>
            <span class="p_text">-{{prediction_value}}-</span>
            {{error}}
             </br>
             </br>
                 <a href="{{url_for('chart')}}">  <button id='btn1' type="sub
mit" class="btn btn-lg btn-primary">Analysis</button></a>
            <div class="col-2"></div>
        </div>
    </div>
</div>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
        integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
        crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popp
er.min.js"
        integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
        crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min
.js"
        integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
        crossorigin="anonymous"></script>
</body>
</html>
```

Login.HTML

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
  <style>
```

```
.submit{
  background-color: #4CAF50; /* Green */
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 2px 2px;
  cursor: pointer;
}

</style>
    <meta charset="utf-8">
    <title></title>
    <link rel="stylesheet" href="../static/css/styles.css">
  </head>
     <body>
        <div class="background">
           <div class="bg"></div>
        <div align="center" style="opacity:1.0;" >
          <form action="{{ url_for('upload')}}">
                <table cellpadding="20" cellspacing="1" border="0" width="70%
">
                    <tr><td colspan="4"></td></tr>
                     <tr><td colspan="4"></td></tr>


                    <tr><td colspan="4" align="center"><h2><b><i>Login HERE</
b></i></h2></td></tr>
                <tr><td width="20%"></td><td><strong><font color="grey">Enter
 Your Username</font></strong></td><td><input type="text" name="txt_uid"></td
><td width="20%"></td></tr>
                <tr><td width="20%"></td><td><strong><font color="grey">Enter
 your Password</font></strong></td> <td><input type="password" name="txt_pass
"></td><td width="20%"></td></tr>
                <tr><td colspan="4" align="center"><input type="submit" class
="submit" value="SUBMIT"></td></tr>

                </table>
```

```
            </form>


            </div>
        </div>
    </body>
</html>
```

Preview.HTML

```
<!DOCTYPE html>
<html lang="en">

<head>

  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="description" content="">
  <meta name="author" content="">

  <title>Fake News Detection</title>

  <!-- Custom fonts for this theme -->
  <link href="../static/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?family=Montserrat:400,700" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?family=Lato:400,700,400italic,700italic" rel="stylesheet" type="text/css">

  <!-- Theme CSS -->
  <link href="../static/css/freelancer.min.css" rel="stylesheet">
    <style>
   #loading {
   background: url('../static/ajax-loader.gif') no-repeat center center;
   position: absolute;
   top: 0;
   left: 59;
   height: 100%;
   width: 86%;
   z-index: 9999999;
```

```
}
   </style>


</head>

<body id="page-top">

  <!-- Navigation -->
  <nav class="navbar navbar-expand-lg bg-secondary text-uppercase fixed-
top" id="mainNav">
    <div class="container">
      <a class="navbar-brand js-scroll-trigger" href="#page-
top">Fake News Detection</a>

  </nav>


  <!-- Contact Section -->
  <section class="page-section" id="contact">
    <div class="container">
     <br>
   <br>
      <!-- Contact Section Heading -->
      <h2 class="text-center text-uppercase text-secondary mb-0">Preview</h2>

      <!-- Icon Divider -->
      <div class="divider-custom">
        <div class="divider-custom-line"></div>
        <div class="divider-custom-icon">
          <i class="fas fa-star"></i>
        </div>
        <div class="divider-custom-line"></div>
      </div>

      <!-- Contact Section Form -->
      <div class="row" style="margin-left:-350px">
        <div class="col-lg-8 mx-auto">
          <!-
- To configure the contact form email address, go to mail/contact_me.php and
update the email address in the PHP file on line 19. -->
    {{ df_view.to_html(classes="table striped",na_rep="-") | safe}}
        </div>
```

```
      </div>

  </section>
 <div class="form-group" style="padding:0px 250px 10px 40px;height:200px">
                    <input style="margin-
left:500px" type="button" onclick="hideLoader()" class="btn btn-
primary" value="Click to Train | Test" />
          <div id="loading" style="display:None;margin-top:552950px"></div>
          </div>


  <!-- Copyright Section -->
  <section class="copyright py-4 text-center text-white">
    <div class="container">

    </div>
  </section>

  <!-- Scroll to Top Button (Only visible on small and extra-
small screen sizes) -->
  <div class="scroll-to-top d-lg-none position-fixed ">
    <a class="js-scroll-trigger d-block text-center text-
white rounded" href="#page-top">
      <i class="fa fa-chevron-up"></i>
    </a>
  </div>


  <!-- Bootstrap core JavaScript -->
  <script src="../static/vendor/jquery/jquery.min.js"></script>
  <script src="../static/vendor/bootstrap/js/bootstrap.bundle.min.js"></scrip
t>

  <!-- Plugin JavaScript -->
  <script src="../static/vendor/jquery-easing/jquery.easing.min.js"></script>

  <!-- Contact Form JavaScript -->
  <script src="../static/js/jqBootstrapValidation.js"></script>
  <script src="../static/js/contact_me.js"></script>

  <!-- Custom scripts for this template -->
  <script src="../static/js/freelancer.min.js"></script>
```

```
  <script type='text/javascript' src='https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js'></script>
   <script type='text/javascript'>

function hideLoader() {
$('#loading').show(0).delay(10000).hide(0,function(){
     alert("Training finished!");
   window.location = "{{url_for('home')}}";
   });
}
   </script>

</body>
</html>
```

Upload.HTML

```
<!DOCTYPE html>
<html lang="en">

<head>

  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="description" content="">
  <meta name="author" content="">

  <title>Fake News Detection</title>

  <!-- Custom fonts for this theme -->
  <link href="../static/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?family=Montserrat:400,700" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?family=Lato:400,700,400italic,700italic" rel="stylesheet" type="text/css">

  <!-- Theme CSS -->
```

```
  <link href="../static/css/freelancer.min.css" rel="stylesheet">


</head>

<body id="page-top">

  <!-- Navigation -->
  <nav class="navbar navbar-expand-lg bg-secondary text-uppercase fixed-
top" id="mainNav">
    <div class="container">
      <a class="navbar-brand js-scroll-trigger" href="#page-
top">Fake News Detection</a>

  </nav>

 <!-- About Section -->
  <section class="page-section bg-primary text-white mb-0" id="about">
    <div class="container">
     <br>
   <br>
      <!-- About Section Heading -->
      <h2 class="text-center text-uppercase text-white">Upload Dataset</h2>

      <!-- Icon Divider -->
      <div class="divider-custom divider-light">
        <div class="divider-custom-line"></div>
        <div class="divider-custom-icon">
          <i class="fas fa-star"></i>
        </div>
        <div class="divider-custom-line"></div>
      </div>

      <!-- About Section Content -->
      <div class="row">
        <div class="col-lg-4 ml-auto" style="margin-right:250px;">
          <form action="http://localhost:5000/preview" name="fs" id="fs" meth
od="post" enctype=multipart/form-data>
                        <br/>
                        <input type="file" name="datasetfile" id="file1" requi
red />
                        <br/>
             <br/>
```

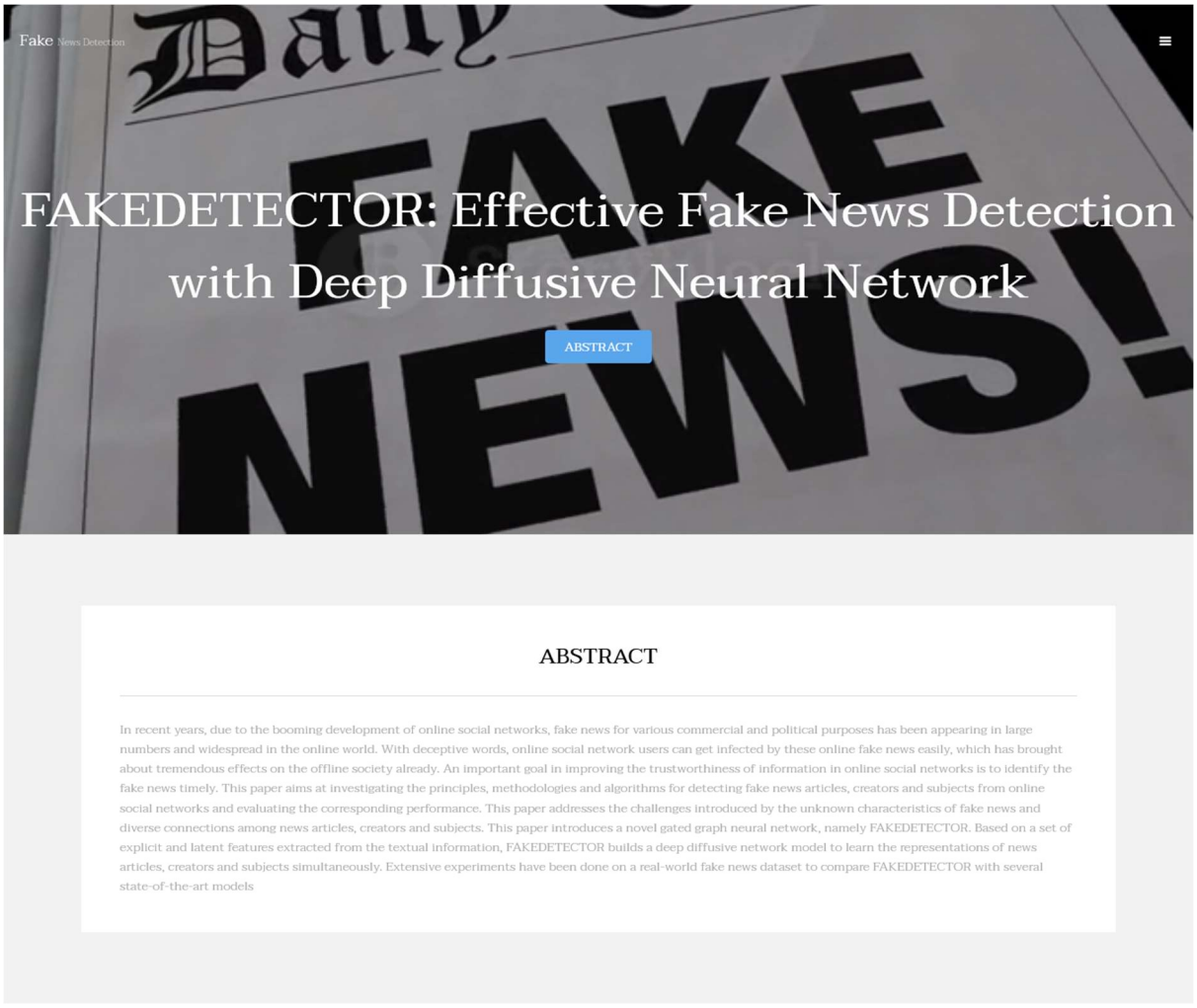
```
            <br/>
                    <input type="submit" style="margin-
right:250px" class="" value="Upload">
                     </form>
      </div>

  </section>



    <div class="scroll-to-top d-lg-none position-fixed ">
    <a class="js-scroll-trigger d-block text-center text-
white rounded" href="#page-top">
      <i class="fa fa-chevron-up"></i>
    </a>
  </div>


  <!-- Bootstrap core JavaScript -->
  <script src="../static/vendor/jquery/jquery.min.js"></script>
  <script src="../static/vendor/bootstrap/js/bootstrap.bundle.min.js"></scrip
t>

  <!-- Plugin JavaScript -->
  <script src="../static/vendor/jquery-easing/jquery.easing.min.js"></script>

  <!-- Contact Form JavaScript -->
  <script src="../static/js/jqBootstrapValidation.js"></script>
  <script src="../static/js/contact_me.js"></script>

  <!-- Custom scripts for this template -->
  <script src="../static/js/freelancer.min.js"></script>

</body> </html>
```

CHAPTER 7
SNAPSHOTS

7.1 GENERAL

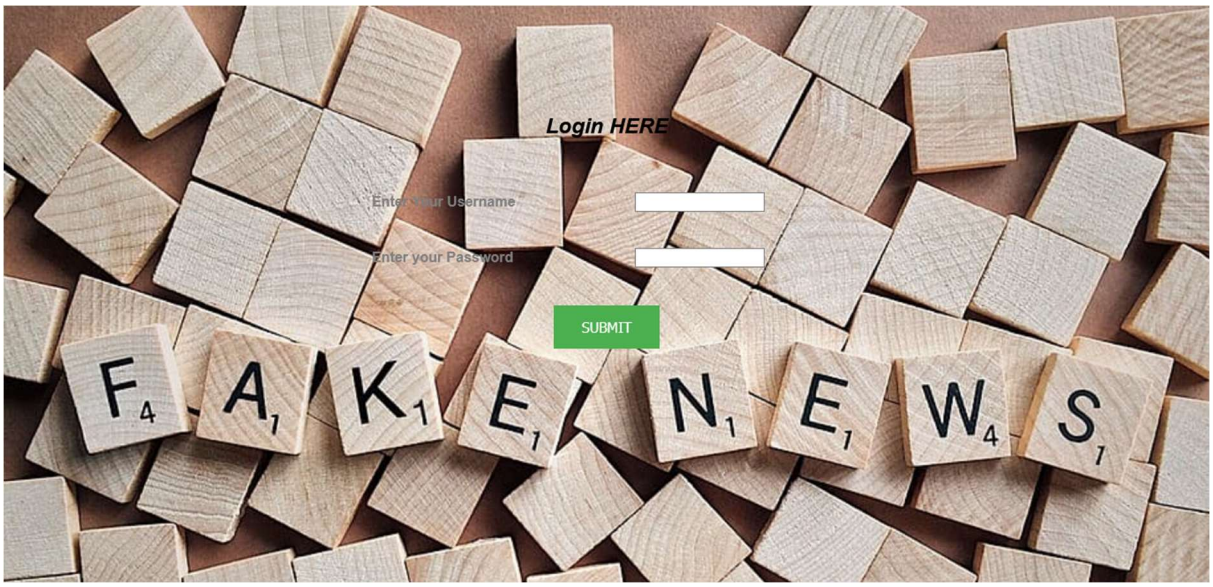
This project is implements like web application using Python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

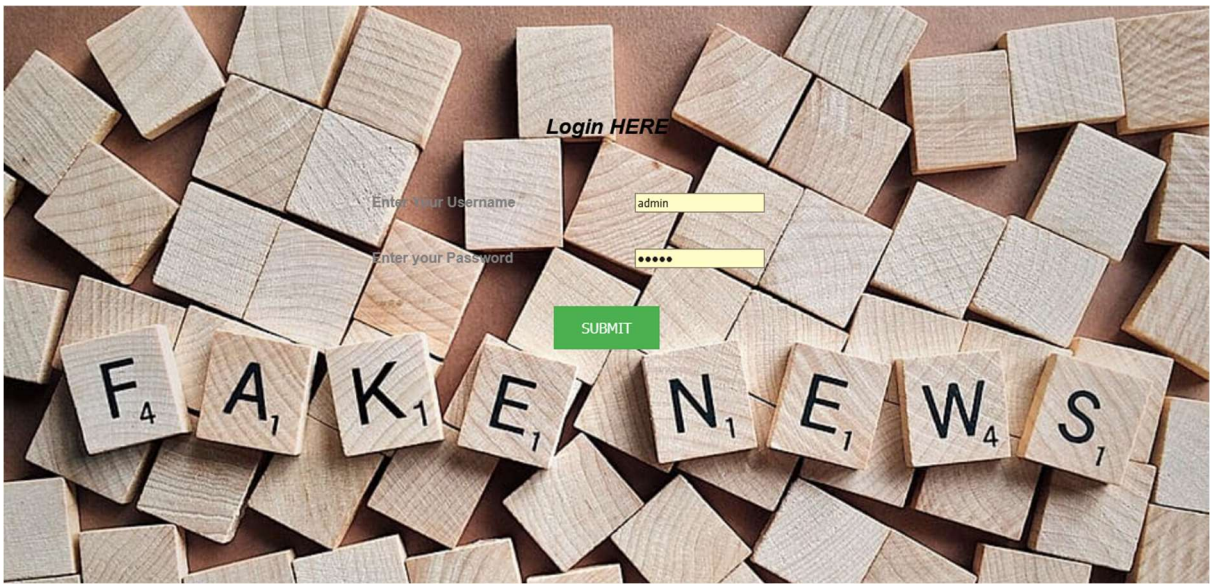
7.2 VARIOUS SNAPSHOTS

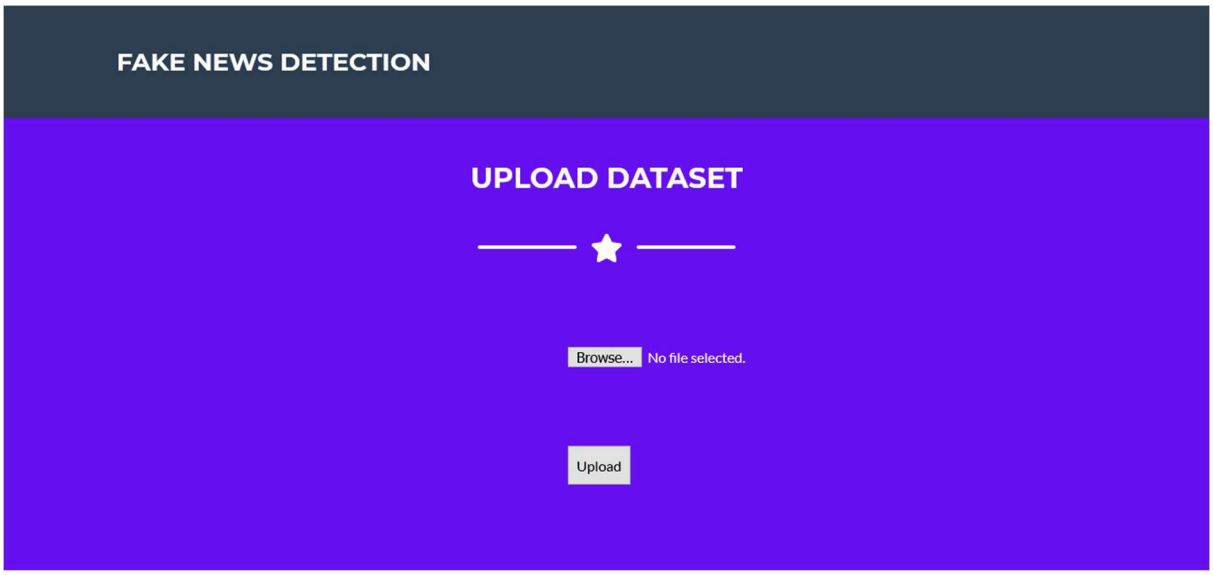


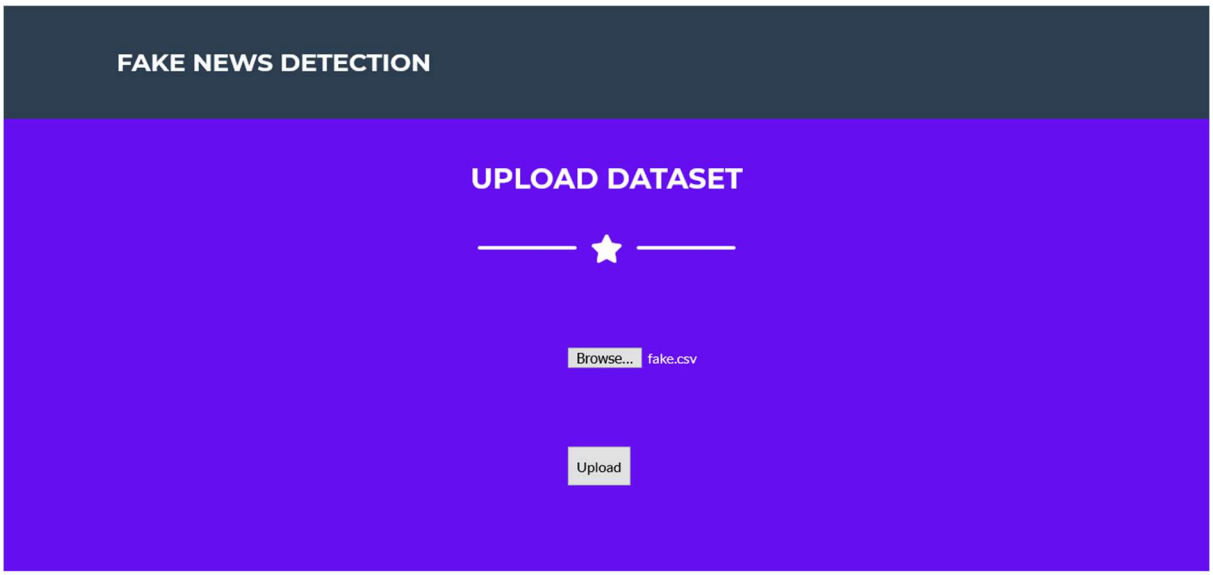
ABSTRACT

In recent years, due to the booming development of online social networks, fake news for various commercial and political purposes has been appearing in large numbers and widespread in the online world. With deceptive words, online social network users can get infected by these online fake news easily, which has brought about tremendous effects on the offline society already. An important goal in improving the trustworthiness of information in online social networks is to identify the fake news timely. This paper aims at investigating the principles, methodologies and algorithms for detecting fake news articles, creators and subjects from online social networks and evaluating the corresponding performance. This paper addresses the challenges introduced by the unknown characteristics of fake news and diverse connections among news articles, creators and subjects. This paper introduces a novel gated graph neural network, namely FAKEDETECTOR. Based on a set of explicit and latent features extracted from the textual information, FAKEDETECTOR builds a deep diffusive network model to learn the representations of news articles, creators and subjects simultaneously. Extensive experiments have been done on a real-world fake news dataset to compare FAKEDETECTOR with several state-of-the-art models


Login HERE
Enter your Username
Enter your Password
SUBMIT
F A K E N E W S


Login HERE
Enter your Username
admin
Enter your Password
SUBMIT
F A K E N E W S

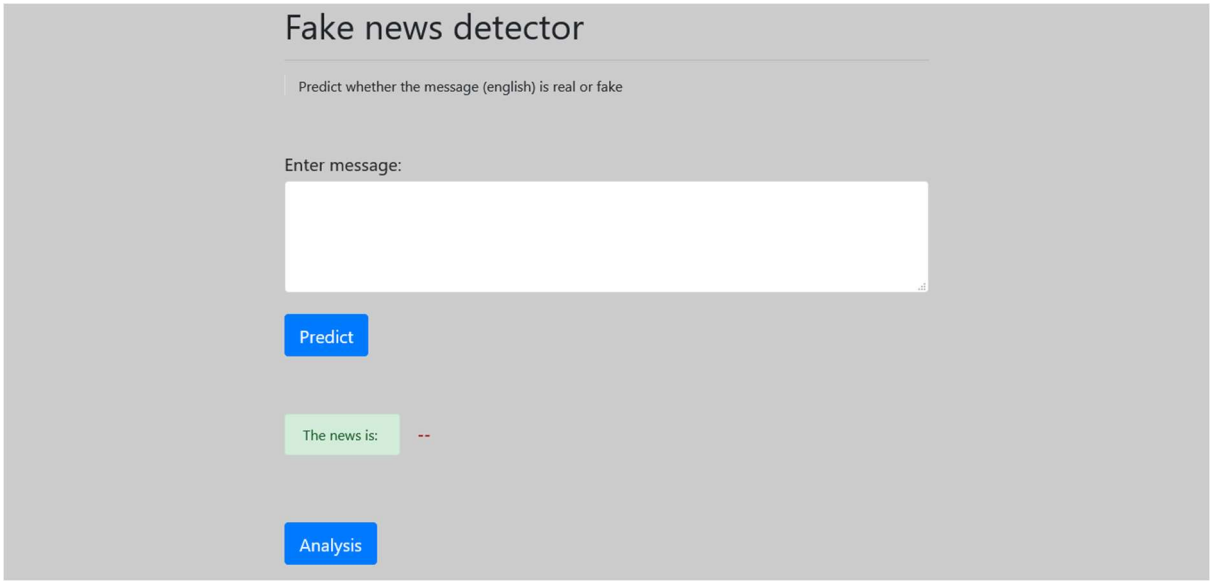

FAKE NEWS DETECTION
UPLOAD DATASET
Browse...
No file selected.
Upload

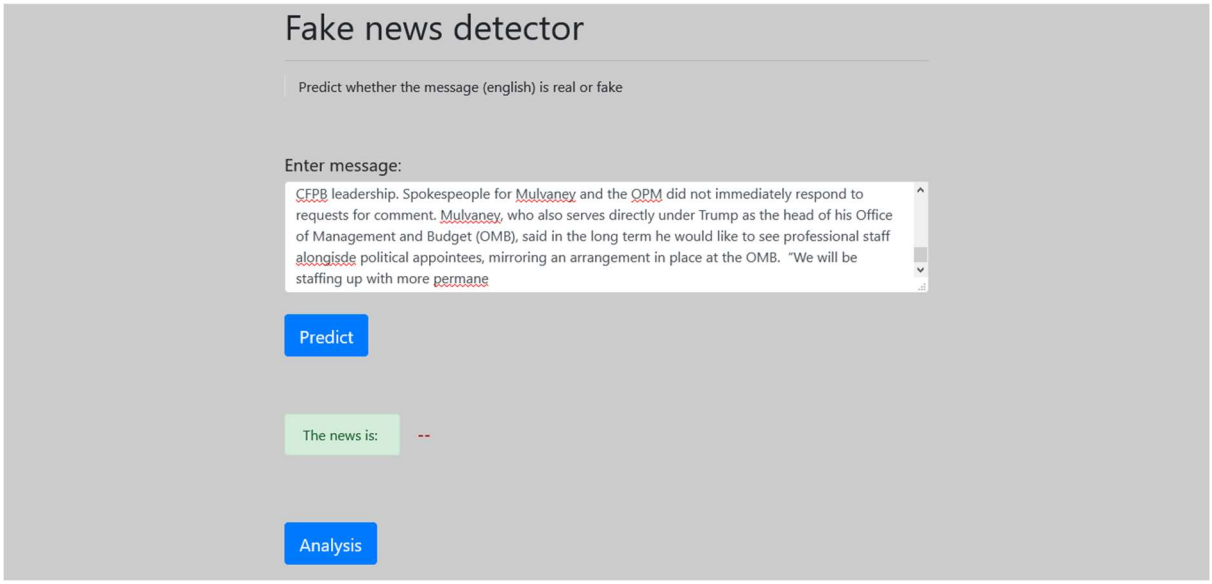

FAKE NEWS DETECTION
UPLOAD DATASET
Browse...
fake.csv
Upload

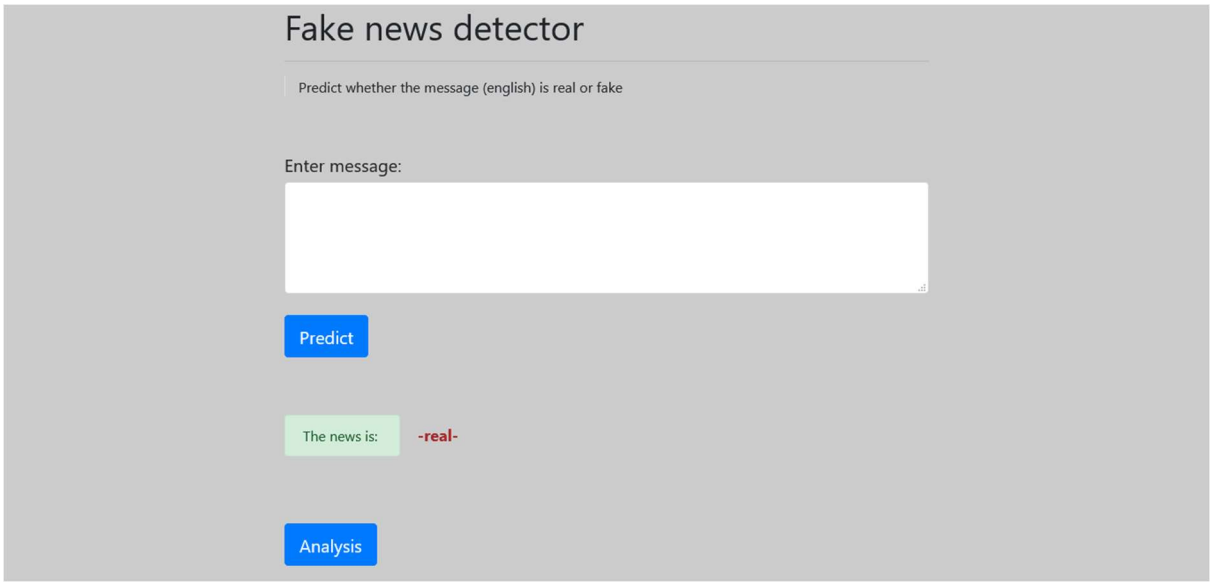
PREVIEW

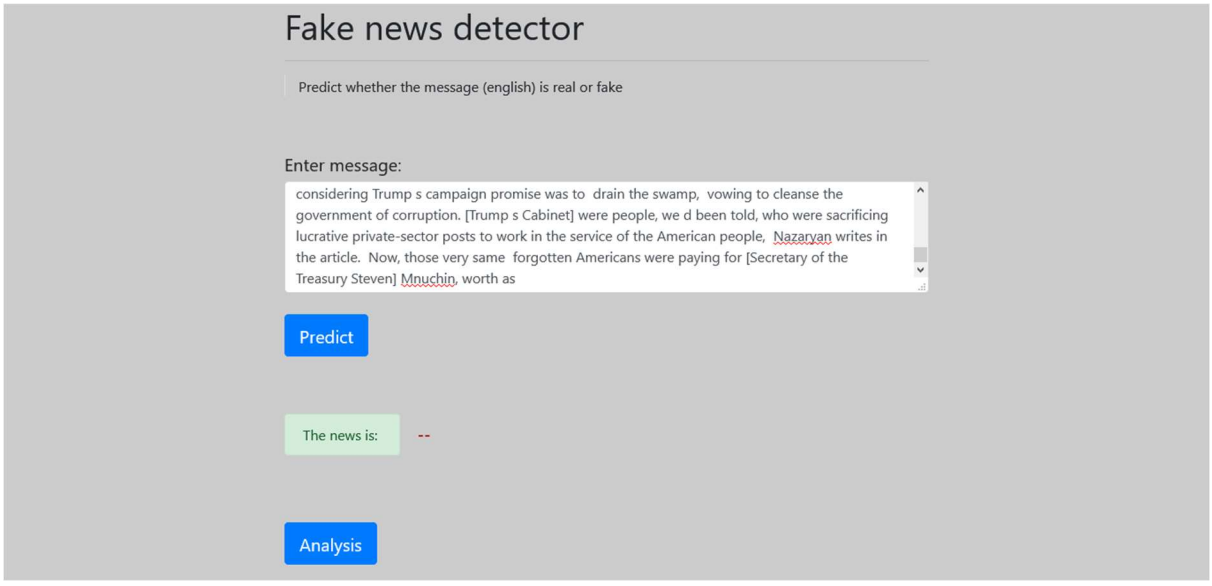
	title	author	text	label
FAKE NEWS DETECTION				
20659	âMoonlight,â âLa La Landâ and What an Epic Oscars Fail Really Says - The New York Times	Manohla Dargis, Wesley Morris and A.O. Scott	After Sunday nightâs when the presenters Warren Beatty and Faye Dunaway announced the wrong best picture, we asked the chief film critics of The New York Times, A. O. Scott and Manohla Dargis, and the Times critic at large, Wesley Morris, to share their reactions. A. O. SCOTT Wow, that âBonnie and Clydeâ sequel was totally nuts. But in its own way, last nightâs spectacle â so relatively smooth, until all of a sudden it was anything but â represents a Hollywood watershed or, at least, like the original	0
			RootsAction.org . Swansonâs books include War Is A	
FAKE NEWS DETECTION				
			a 2015 and 2016 Nobel Peace Prize Nominee. \nFollow him on Twitter: @davidcnswanson and FaceBook . \nHelp support DavidSwanson.org, WarIsACrime.org, and TalkNationRadio.org by clicking here: http://davidswanson.org/donate .	

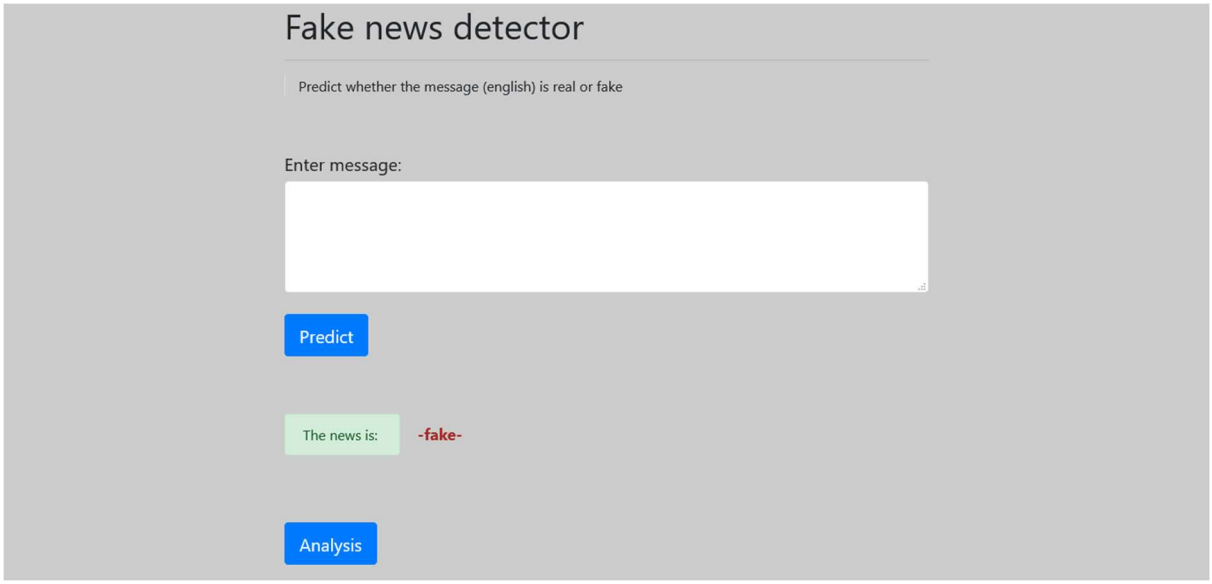
Click to Train | Test


Fake news detector
Predict whether the message (english) is real or fake
Enter message:
Predict
The news is:
--
Analysis

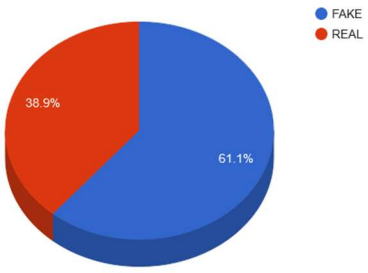

Fake news detector
Predict whether the message (english) is real or fake
Enter message:
CFPB leadership. Spokespeople for Mulvaney and the OPM did not immediately respond to requests for comment. Mulvaney, who also serves directly under Trump as the head of his Office of Management and Budget (OMB), said in the long term he would like to see professional staff alongisde political appointees, mirroring an arrangement in place at the OMB. "We will be staffing up with more permane
Predict
The news is:
--
Analysis


Fake news detector
Predict whether the message (english) is real or fake
Enter message:
Predict
The news is:
-real-
Analysis


Fake news detector
Predict whether the message (english) is real or fake
Enter message:
considering Trump s campaign promise was to drain the swamp, vowing to cleanse the government of corruption. [Trump s Cabinet] were people, we d been told, who were sacrificing lucrative private-sector posts to work in the service of the American people, Nazaryan writes in the article. Now, those very same forgotten Americans were paying for [Secretary of the Treasury Steven] Mnuchin, worth as
Predict
The news is:
--
Analysis



Fake News Detection (pie chart analysis)



Future

CONCLUSION AND FUTURE WORK

The introduced GDU model also works for both the news subjects and creator nodes in the network. When applying the GDU to model the states of the subject/creator nodes with two input only, the remaining input port can be assigned with a default value (usually vector 0). Based on the GDU, we can denote the overall architecture of the FAKEDETECTOR as shown in Figure 4, where the lines connecting the GDUs denote the data flow among the unit models. In the following section, we will introduce how to learn the parameters involved in the FAKEDETECTOR model for concurrent credibility inference of multiple nodes.

HOME

CHAPTER 8
SOFTWARE TESTING

8.1 GENERAL

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, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

8.2 DEVELOPING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

8.3 Types of Tests

8.3.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input 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 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.

8.3.2 Functional test

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.

8.3.3 System Test

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.

8.3.4 Performance Test

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

8.3.5 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.

8.3.6 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.

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updating process

8.3.7 Build the test plan

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

8.4 Unit Testing:

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 work properly.
- 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.

8.5 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 encountered.

8.6 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 encountered.

CHAPTER 9
APPLICATION

9.1 GENERAL

By resolving the optimization functions, we will be able to learn the variables involved in the framework. In this paper, we propose to train the framework with the back-propagation algorithm. For the news articles, creators and subjects in the testing set, their predicted credibility labels will be outputted as the final result. More information about the experimental studies of FAKEDETECTOR on real-world datasets is available in our full-version of this paper at.

9.2 FUTURE ENHANCEMENT

Furthermore, the model seems to be relatively unphased by the exclusion of certain "giveaway" topic words in the training set, as it is able to pick up on trigrams that are less specific to a given topic, if need be. As such, this seems to be a really good start on a tool that would be useful to augment human's ability to detect Fake News.

CHAPTER 10
CONCLUSION & REFERENCES

10.1 CONCLUSION

The main contribution of this project is support for the idea that machine learning could be useful in a novel way for the task of classifying fake news. Our findings show that after much pre-processing of relatively small dataset, a simple CNN is able to pick up on a diverse set of potentially subtle language patterns that a human may (or may not) be able to detect. Many of these language patterns are intuitively useful in a human's manner of classifying fake news. Some such intuitive patterns that our model has found to indicate fake news include generalizations, colloquialisms and exaggerations.

Likewise, our model looks for indefinite or inconclusive words, referential words, and evidence words as patterns that characterize real news. Even if a human could detect these patterns, they are not able to store as much information as a CNN model, and therefore, may not understand the complex relationships between the detection of these patterns and the decision for classification.

10.2 REFERENCES

[1] Great moon hoax. https://en.wikipedia.org/wiki/Great Moon Hoax. [Online; accessed 25 September-2017].

[2] L. Akoglu, R. Chandy, and C. Faloutsos. Opinion fraud detection in online reviews by network effects. In ICWSM, 2013.

[3] H. Allcott and M. Gentzkow. Social media and fake news in the 2016 election. Journal of Economic Perspectives, 2017.

[4] J. Chung, C¸ . G¨ulc¸ehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555, 2014.

[5] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Zhao. Detecting and characterizing social spam campaigns. In IMC, 2010.

[6] Y. Kim. Convolutional neural networks for sentence classification. In EMNLP, 2014.

[7] S. Lin, Q. Hu, J. Zhang, and P. Yu. Discovering Audience Groups and Group-Specific Influencers. 2015.

[8] Y. Teng, C. Tai, P. Yu, and M. Chen. modelling and utilizing dynamic influence strength for personalized promotion. In ASONAM, 2015.

[9] S. Xie, G. Wang, S. Lin, and P. Yu. Review spam detection via temporal pattern discovery. In KDD, 2012.

[10] S. Xie, G. Wang, S. Lin, and P. Yu. Review spam detection via time series pattern discovery. In WWW, 2012.

[11] Y. Yang, L. Zheng, J. Zhang, Q. Cui, Z. Li, and P. Yu. TI-CNN: convolutional neural networks for fake news detection. CoRR, abs/1806.00749, 2018.

[12] Q. Zhan, J. Zhang, S. Wang, P. Yu, and J. Xie. Influence maximization across partially aligned heterogenous social networks. In PAKDD. 2015.

[13] J. Zhang, B. Dong, and P. Yu. Fakedetector: Effective fake news detection with deep diffusive neural network. CoRR, abs/1805.08751, 2018.