ABSTRACT:

Social media for news consumption is a double-edged sword. On the one hand, its low cost, easy access, and rapid dissemination of information lead people to seek out and consume news from social media. On the other hand, it enables the wide spread of “fake news”, i.e., low quality news with intentionally false information. The extensive spread of fake news has the potential for extremely negative impacts on individuals and society. Therefore, fake news detection on social media has recently become an emerging research that is attracting tremendous attention. Fake news detection on social media presents unique characteristics and challenges that make existing detection algorithms from traditional news media ineffective or not applicable. First, fake news is intentionally written to mislead readers to believe false information, which makes it difficult and nontrivial to detect based on news content; therefore, we need to include auxiliary information, such as user social engagements on social media, to help make a determination. Second, exploiting this auxiliary information is challenging in and of itself as users’ social engagements with fake news produce data that is big, incomplete, unstructured, and noisy. Because the issue of fake news detection on social media is both challenging and relevant, we conducted this survey to further facilitate research on the problem. In this survey, we present a comprehensive review of detecting fake news on social media, including fake news characterizations on psychology and social theories, existing algorithms from a data mining perspective, evaluation metrics and representative datasets. We also discuss related research areas, open problems, and future research directions for fake news detection on social media.

INTRODUCTION:

As an increasing amount of our lives is spent interacting online through social media platforms, more and more people tend to seek out and consume news from social media rather than traditional news organizations. The reasons for this change in consumption behaviours are inherent in the nature of these social media platforms: (i) it is often more timely and less expensive to consume news on social media compared with traditional news media, such as newspapers or television; and (ii) it is easier to further share, comment on, and discuss the news with friends or other readers on social media. For example, 62 percent of U.S. adults get news on social media in 2016, while in 2012, only 49 percent reported seeing news on social media [1]. It was also found that social media now outperforms television as the major news source [2]. Despite the advantages provided by social media, the quality of news on social media is lower than traditional news organizations. However, because it is cheap to provide news online and much faster and easier to disseminate through social media, large volumes of fake news, i.e., those news articles with intentionally false information, are produced online for a variety of purposes, such as ﬁnancial and political gain. It was estimated that over 1 million tweets are related to fake news “Pizzagate” [3] by the end of the presidential election. Given the prevalence of this new phenomenon, “Fake news” was even named the word of the year by the Macquarie dictionary in 2016.

The extensive spread of fake news can have a serious negative impact on individuals and society. First, fake news can break the authenticity balance of the news ecosystem. For example, it is evident that the most popular fake news was even more widely spread on Facebook than the most popular authentic mainstream news during the U.S. 2016 president election [4]. Second, fake news intentionally persuades consumers to accept biased or false beliefs. Fake news is usually manipulated by propagandists to convey political messages or inﬂuence. For example, some report shows that Russia has created fake accounts and social bots to spread false stories [5]. Third, fake news changes the way people interpret and respond to real news. For example, some fake news was just created to trigger people’s distrust and make them confused, impeding their abilities to differentiate what is true from what is not [6]. To help mitigate the negative effects caused by fake news–both to benefit the public and the news ecosystem–It’s critical that we develop methods to automatically detect fake news on social media.

Detecting fake news on social media poses several new and challenging research problems. Though fake news itself is not a new problem–nations or groups have been using the news media to execute propaganda or influence operations for centuries–the rise of web-generated news on social media makes fake news a more powerful force that challenges traditional journalistic norms. There are several characteristics of this problem that make it uniquely challenging for automated detection. First, fake news is intentionally written to mislead readers, which makes it nontrivial to detect simply based on news content. The content of fake news is rather diverse in terms of topics, styles and media platforms, and fake news attempts to distort truth with diverse linguistic styles while simultaneously mocking true news. For example, fake news may cite true evidence within the incorrect context to support a non-factual claim. Thus, existing hand-crafted and data-specific textual features are generally not sufficient for fake news detection. Other auxiliary information must also be applied to improve detection, such as knowledge base and user social engagements. Second, exploiting this auxiliary information actually leads to another critical challenge: the quality of the data itself. Fake news is usually related to newly emerging, time-critical events, which may not have been properly verified by existing knowledge bases due to the lack of corroborating evidence or claims. In addition, users’ social engagements with fake news produce data that is big, incomplete, unstructured, and noisy. Effective methods to differentiate credible users, extract useful post features and exploit network interactions are an open area of research and need further investigations.

In this article, we present an overview of fake news detection and discuss promising research directions. The key motivations of this survey are summarized as follows:

* Fake news on social media has been occurring for several years; however, there is no agreed upon deﬁnition of the term “fake news”. To better guide the future directions of fake news detection research, appropriate clariﬁcations are necessary.
* Social media has proved to be a powerful source for fake news dissemination. There are some emerging patterns that can be utilized for fake news detection in social media. A review on existing fake news detection methods under various social media scenarios can provide a basic understanding on the state-of-the-art fake news detection methods.
* Fake news detection on social media is still in the early age of development, and there are still many challenging issues that need further investigations. It is necessary to discuss potential research directions that can improve fake news detection and mitigation capabilities.

To facilitate research in fake news detection on social media, in this survey we will review two aspects of the fake news detection problem: characterization and detection. As shown in Figure 1, we will first describe the background of the fake news detection problem using theories and properties from psychology and social studies; then we present the detection approaches. Our major contributions of this survey are summarized as follows:

* We discuss the narrow and broad definitions of fake news that cover most existing definitions in the literature and further present the unique characteristics of fake news on social media and its implications compared with the traditional media;
* We give an overview of existing fake news detection methods with a principled way to group representative methods into different categories; and
* We discuss several open issues and provide future directions of fake news detection in social media.

LITERATURE REVIEW:

While many social media users are very much real, those who are malicious and out to spread lies may or may not be real people. There are three main types of fake news contributors: social bots, trolls, and cyborg users (Shu et al., 2017). Since the cost to create social media accounts is very low, the creation of malicious accounts is not discouraged. If a social media account is being controlled by a computer algorithm, then it is referred to as a social bot. A social bot can automatically generate content and even interact with social media users. Social bots may or may not always be harmful but it entirely depends on how they are programmed. If a social bot is designed with the sole purpose of causing harm, such as spreading fake news in social media, then they can be very malicious entities and contribute greatly to the creation of fake news. For example, “studies shows that social bots distorted the 2016 US presidential election discussions on a large scale, and around 19 million bot accounts tweeted in support of either Trump or Clinton in the week leading up to the election day,” which demonstrates how influential social bots can be on social media (Shu et al., 2017).

However, fake humans are not the only contributors to the dissemination of false information; real humans are very much active in the domain of fake news. As implied, trolls are real humans who “aim to disrupt online communities” in hopes of provoking social media users into an emotional response (Shu et al., 2017).

For instance, there has been evidence that claims “1,000 Russian trolls were paid to spread fake news on Hilary Clinton,” which reveals how actual people are performing information manipulation in order to change the views of others (Shu et al., 2017).

The main goal of trolling is to resurface any negative feelings harvested in social media users, such as fear and even anger, so that users will develop strong emotions of doubt and distrust (Shu et al., 2017). When a user has doubt and distrust in their mind, they won’t know what to believe and may start doubting the truth and believing the lies instead.

While contributors of fake news can be either real or fake, what happens when it’s a blend of both? Cyborg users are a combination of “automated activities with human input” (Shu et al., 2017).

The accounts are typically registered by real humans as a cover, but use programs to perform activities in social media. What makes cyborg users even more powerful is that they are able to switch the “functionalities between human and bot,” which gives them a great opportunity to spread false information (Shu et al., 2017).

ARCHITECTURE:

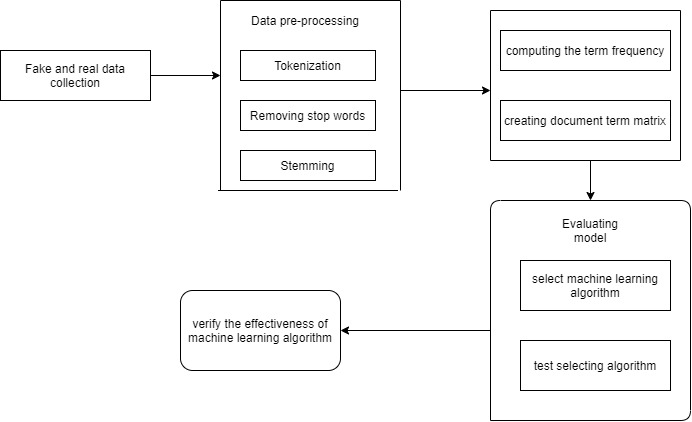


Figure: Architecture of fake news detection

SYSTEM DESIGN:

Introduction to UML:

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspectives. Each view is defined by a set of diagram, which is as follows:

1. User Model View
   1. This view represents the system from the users’ perspective.
   2. The analysis representation describes a usage scenario from the end-users’ perspective.
2. Structural Model View
   1. In this model, the data and functionality are arrived from inside the system.
   2. This model view models the static structures.
3. Behavioural Model View

It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

4. Implementation Model View

In this view, the structural and behavioural as parts of the system are represented as they are to be built.

5. Environmental Model View

In this view, the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

UML diagrams:

Use case Diagram:

To model a system, the most important aspect is to capture the dynamic behaviour. To clarify a bit in detail, dynamic behaviour means the behaviour of the system when it is running /operating.

So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consisting of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So to model the entire system numbers of use case diagrams are used.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analysed to gather its functionalities use cases are prepared and actors are identified. In brief, the purposes of use case diagrams can be as follows:

1. Used to gather requirements of a system.
2. Used to get an outside view of a system.
3. Identify external and internal factors influencing the system.
4. Show the interacting among the requirements are actors.

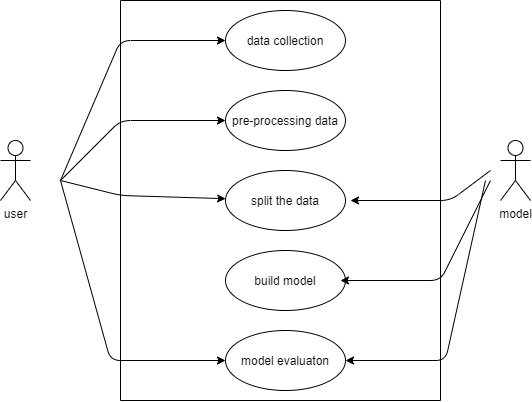


Figure : Use case for fake news detection

Class Diagram:

Class diagrams are the main building blocks of every object oriented method. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is based on OOPs, so as the class diagram has appropriate structure to represent the classes, inheritance, relationships, and everything that OOPs have in its context. It describes various kinds of objects and the static relationship between them.

The main purpose to use class diagrams are:

* This is the only UML which can appropriately depict various aspects of OOPs concept.
* Proper design and analysis of application can be faster and efficient.
* It is based on deployment and component diagrams.

Each class is represented by a rectangle having a subdivision of three compartments name, attributes and operation.

There are three types of modifiers which are used to decide the visibility of attributes and operations.

* + is used for public visibility (for everyone)
* # is used for protected visibility (for friend and derived)
* – is used for private visibility (for me)

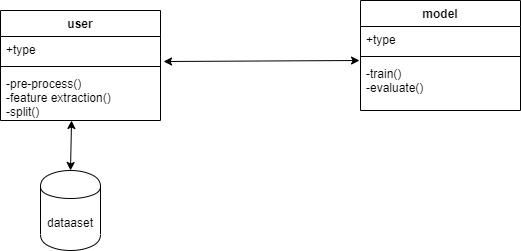


Figure: class diagram for fake news detection

Sequence diagram:

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.

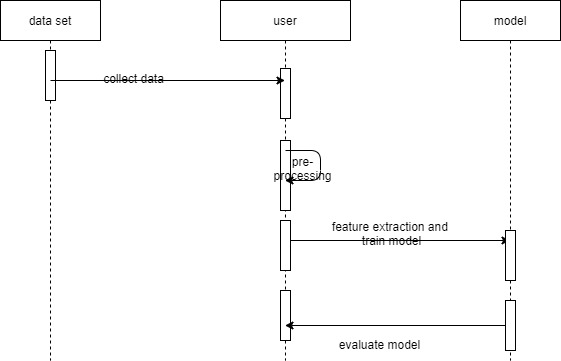


Figure: sequence diagram for fake news detection

Activity Diagram:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all types of flow control by using different elements such as fork, join, etc.

## Purpose of Activity Diagrams

The basic purposes of activity diagrams are similar to the other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as :

* Draw the activity flow of a system.
* Describe the sequence from one activity to another.
* Describe the parallel, branched and concurrent flow of the system.

Activity Diagram Notations :

1. Initial State – The starting state before an activity takes place is depicted using the initial state.

UML-State-Diagram

Symbol 1:– notation for initial state or start state

A process can have only one initial state unless we are depicting nested activities. We use a black filled circle to depict the initial state of a system. The Initial State from the UML Activity Diagram marks the entry point and the initial Activity State.

For example – Here the initial state is the state of the system before the application is opened.



 initial state symbol being used

1. Action or Activity State – An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically any action or event that takes place is represented using an activity.

UML-Activity-Diagram

Symbol 2 – notation for an activity state

For example – Consider the previous example of opening an application opening the application is an activity state in the activity diagram.

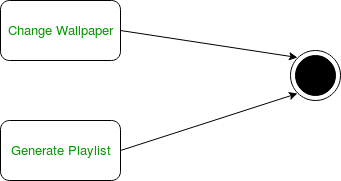
  
activity state symbol being used

1. Action Flow or Control flows – Action flow or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another.

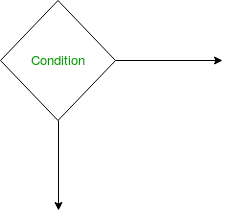
UML-Object-Diagram  
Symbol 3 – notation for control Flow

An activity state can have multiple incoming and outgoing action flows. We use a line with an arrow head to depict a Control Flow. If there is a constraint to be adhered to while making the transition it is mentioned on the arrow.

1. Arrows:-Consider the example – Here both the states transit into one final state using action flow symbols i.e. arrows.

  
Symbol 4 – using action flows for transitions

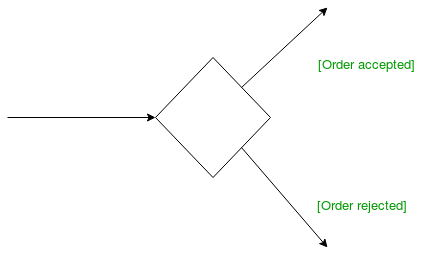
1. Decision node and Branching – When we need to make a decision before deciding the flow of control, we use the decision node.

  
Symbol 5 – notation for decision node

The outgoing arrows from the decision node can be labelled with conditions or guard expressions. It always includes two or more output arrows.

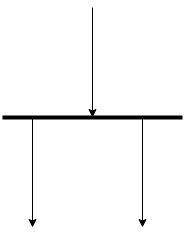
1. Guards – A Guard refers to a statement written next to a decision node on an arrow sometimes within square brackets.

The statement must be true for the control to shift along a particular direction. Guards help us know the constraints and conditions which determine the flow of a process.



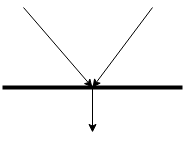
Symbol 6– guards being used next to a decision node

5. Fork – Fork nodes are used to support concurrent activities.

  
Symbol 7 – fork notation

When we use a fork node when both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts. Both parts need to be executed in case of a fork statement.  
We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent activity state and outgoing arrows towards the newly created activities.  
For example: In the example below, the activity of making coffee can be split into two concurrent activities and hence we use the fork notation.

1. Join – Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge.

  
Symbol 8 – join notation

1. Final State or End State – The state which the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final states.

UML-State-Diagram  
Symbol 9 – notation for final state

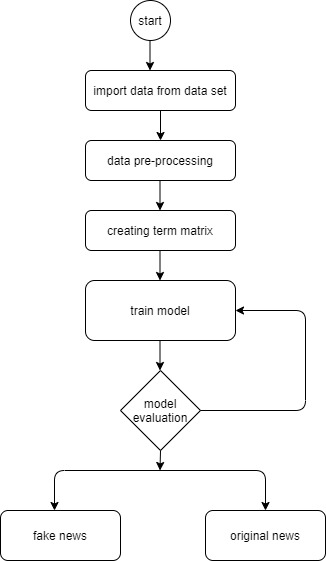


Figure: activity diagram for fake news detection

TECHNOLOGY DESCRIPTION:

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.

Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

* Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
* Win 9x/NT/2000
* Macintosh (Intel, PPC, 68K)
* OS/2
* DOS (multiple versions)
* PalmOS
* Nokia mobile phones
* Windows CE
* Acorn/RISC OS
* BeOS
* Amiga
* VMS/OpenVMS
* QNX
* VxWorks
* Psion
* Python has also been ported to the Java and .NET virtual machines

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/>

You can download Python documentation from <https://www.python.org/doc/>. The documentation is available in HTML, PDF, and PostScript formats.

Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms −

Unix and Linux Installation

Here are the simple steps to install Python on Unix/Linux machine.

* Open a Web browser and go to <https://www.python.org/downloads/>.
* Follow the link to download zipped source code available for Unix/Linux.
* Download and extract files.
* Editing the *Modules/Setup* file if you want to customize some options.
* run ./configure script
* make
* make install

This installs Python at standard location */usr/local/bin* and its libraries at */usr/local/lib/pythonXX* where XX is the version of Python.

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.msi* file 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.

IMPLEMENTATION:

Import data:

We import data from the dataset in the format of csv. The data consist of real news and fake news’s.

Data pre-processing:

After import data then we have to pre-processing the data, in the pre-processing involve below steps,

None of the datasets contains missing values therefore no cleaning required.

If it is find null values fill that null values data to next or average value in the data set

Remove the stop words in the data and stemming.

Feature extraction:

Text feature extraction that extracts text information is an extraction to represent a text message, it is the basis of a large number of text processing. The basic unit of the feature is called text features. Selecting a set of features from some effective ways to reduce the dimension of feature space, the purpose of this process is called feature extraction. During feature extraction, uncorrelated or superfluous features will be deleted. As a method of data preprocessing of learning algorithm, feature extraction can better improve the accuracy of learning algorithm and shorten the time. Selection from the document part can reflect the information on the content words, and the calculation of weight is called the text feature extraction. Common methods of text feature extraction include filtration, fusion, mapping, and clustering method. Traditional methods of feature extraction require handcrafted features. To hand-design an effective feature is a lengthy process, and deep learning can be aimed at new applications and quickly acquire new effective characteristic representation from training data. We use extract features from data here is bag f words technique and matrix vectorization.

Classification:

Now, you’ll need to define the tags that you will use for the text classifier. These are the categories or buckets that your model will make predictions for:

While defining your tags, avoid using tags that are ambiguous or overlapping as this can cause confusion to your classifier and affect its accuracy.

Also, it’s a good idea to structure your tags and build a hierarchical text classification process. This means that you should organize your tags according to their semantic relations.

**Prediction:**

Once we create classifiers then we have to test our testing data. Once the predictions are good enough, you can use the classifier to analyze and categorize new unseen text.

Used libraries:

Numpy:

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided.

Operating using numpy:

Using NumPy, a developer can perform the following operations −

* Mathematical and logical operations on arrays.
* Fourier transforms and routines for shape manipulation.
* Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

Pandas:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Warning:**

Warnings are provided to warn the developer of situations that aren’t necessarily exceptions. Usually, a warning occurs when there is some obsolete of certain programming elements, such as keyword, function or class, etc. A warning in a program is distinct from an error. Python program terminates immediately if an error occurs. Conversely, a warning is not critical. It shows some message, but the program runs. The warn() function defined in the ‘warning‘ module is used to show warning messages. The warning module is actually a subclass of Exception which is a built-in class in Python.

Nltk:

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

SkLearn:

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

* NumPy: Base n-dimensional array package
* SciPy: Fundamental library for scientific computing
* Matplotlib: Comprehensive 2D/3D plotting
* IPython: Enhanced interactive console
* Sympy: Symbolic mathematics
* Pandas: Data structures and analysis

SOFTWARE TESTING

GENERAL

### Testing is an important component in software life cycle. It aids in finding and repairing uncovered errors so that the software does not pose any problems to the vendor. Since the system is developed using an object oriented approach, class testing is done at unit level and functional testing is done at system level.

Testing objectives include:

* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has a high probability of finding an as yet undiscovered error.

Testing is a schedule process carried out by the software development team to capture all the possible errors, missing operations and also a complete verification to verify objective are met and user requirement are satisfied. The design of tests for software and other engineering products can be as challenging as the initial design to the product itself.

# TESTING TYPES

A software engineering product can be tested in one of two ways:

* Black box testing
* White box testing

BLACK BOX TESTING

Knowing the specified function that a product has been designed to perform, determine whether each function is fully operational.

# WHITE BOX TESTING

Knowing the internal workings of a software product determine whether the internal operation implementing the functions perform according to the specification, and all the internal components have been adequately exercised.

# TESTING STRATEGIES

Four Testing Strategies that are often adopted by the software development team include:

* Unit Testing
* Functional Testing
* Performance Test
* Validation Testing

# UNIT TESTING

We adopt white box testing when using this testing technique. This testing was carried out on individual components of the software that were designed. Each individual module was tested using this technique during the coding phase. Every component was checked to make sure that they adhere strictly to the specifications spelt out in the data flow diagram and ensure that they perform the purpose intended for them.

All the names of the variables are scrutinized to make sure that they are truly reflected of the element they represent. All the looping mechanisms were verified to ensure that they were as decided. Beside these, we trace through the code manually to capture syntax errors and logical errors.

FUNCTIONAL TESTING

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

PERFORMANCE TEST

The Performance test ensures that the output is 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.

# VALIDATION TESTING

Software testing and validation is achieved through a series of black box tests that demonstrate conformity with requirements. A test procedure defines specific test cases that will be used to demonstrate conformity with requirements. Both, the plan and the procedure are designed to ensure that all functional requirements are achieved, documentation is correct and other requirements are met. After each validation test case has been conducted, one of the two possible conditions exists.

The function or performance characteristics conform to specification and are accepted. A deviation from specification is uncovered and a deficiency list is created. The deviation or error discovered at this stage in project can rarely be corrected prior to scheduled completion. It is necessary to negotiate with the customer to establish a method for resolving deficiencies.

GUIDELINES FOR DEVELOPING TEST CASES

* Describe which feature or service your test attempts to cover
* If the test case is based on a use case it is a good idea to refer to the use case name. Remember that the use cases are the source of test cases. In theory the software is supposed to match the use cases not the reverse. As soon as you have enough use cases , go ahead and write the test plan for that piece
* Specify what you are testing and which particular feature. Then specify what you are going to do to test the feature and what you expect to happen.
* Test the normal use of the object’s methods. Test the abnormal but reasonable use of the object’s methods.
* Test the abnormal but unreasonable use of the object’s methods.
* Test the boundary conditions. Also specify when you expect error dialog boxes, when you expect some default event, and when functionality till is being defined.
* Test object’s interactions and the messages sent among them. If you have developed sequence diagrams, they can assist you in this process
* when the revisions have been made, document the cases so they become the starting bases for the follow- up test
* Attempting to reach agreement on answers generally will raise other what-if questions. Add these to the list and answer them, repeat the process until the list is stabilized, then you need not add any more questions.

TEST CASES

A test case in software engineering is a set of conditions or variables under which a tester will determine if a requirement or use case upon an application is partially or fully satisfied. It may take many test cases to determine that a requirement is fully satisfied. The following steps are to be followed to design the test cases.

* Each test case should be uniquely identified and explicitly associated with the class to be tested.
* The purpose of the test should be stated.
* Each test case should be uniquely identified and explicitly associated with the class to be tested.
* The purpose of the test should be stated.

Test Cases usually have the following components.

* Test Case Summary
* Initial Condition
* Steps to run the test case
* Expected behavior/outcome

RESULTS:

CONCLUSION:

With the increasing popularity of social media, more and more people consume news from social media instead of traditional news media. However, social media has also been used to spread fake news, which has strong negative impacts on individual users and broader society. In this article, we explored the fake news problem by reviewing existing literature in two phases: characterization and detection. In the characterization phase, we introduced the basic concepts and principles of fake news in both traditional media and social media. In the detection phase, we reviewed existing fake news detection approaches from a data mining perspective, including feature extraction and model construction. We also further discussed the datasets, evaluation metrics, and promising future directions in fake news detection research and expand the field to other applications.

REFRENCES:

[1] Sadia Afroz, Michael Brennan, and Rachel Greenstadt. Detecting hoaxes, frauds, and deception in writing style online. In ISSP’12.

[2] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. Technical report, National Bureau of Economic Research, 2017.

[3] Solomon E Asch and H Guetzkow. Eﬀects of group pressure upon the modiﬁcation and distortion of judgments. Groups, leadership, and men, pages 222–236, 1951.

[4] Meital Balmas. When fake news becomes real: Combined exposure to multiple news sources and political attitudes of ineﬃcacy, alienation, and cynicism. Communication Research, 41(3):430–454, 2014.

[5] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In IJCAI’07.

[6] Alessandro Bessi and Emilio Ferrara. Social bots distort the 2016 us presidential election online discussion. First Monday, 21(11), 2016.