****

**Department of Computer Science**

**COMP4300 - Graduation Project**

**Final Report 2022/2023**

**Title: Fake News Detection**

**Supervisor:**

**Iyad Jaber**



**Group Members:**

· **Laith Isbaitan – 1190628**

· **Adam Saleem – 1190656**

· **Ibrahim AbuAmeera – 1191113**

**Section – B**

**Title of Project:**

**Project No:**

**Supervisor:**

K**ey Areas:**

**Section – C**

**Student Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Date Submitted:**

**First Supervisor Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**First Supervisor Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Date Approved: \_\_\_\_ / \_\_\_\_ / \_\_\_\_\_**

# Acknowledgments

*First, I would like to thank Dr. Iyad Jaber a lot for his helpful, kind, patience, and for making everything simple. He was always inspiring, giving us some much needed feedback and encouraging us to move forward.*

*Also, we would like to thank Dr. Radi Jarrar, because he always had time to answer some of our machine learning questions, and steer us in the needed technologies when we needed it.*

# ABSTRACT

Fake news and misinformation continue to grow faster and in different variations over time, and due to the COVID-19 pandemic, misinformation in social media sites has increased. In this paper, we review projects that tackled this problem and their different approaches, as well as our solution proposal to this epidemic of fake news. This project's main idea is to combine the summarization technique of TextRank, with the data collection of Newspaper3k, and to transform the simple Naive Bayes model into a complex Graph-based detection model with Bayesian Networks. With this, we can take advantage of graphs and their ability to form relations that otherwise would go unnoticed. We also propose taking advantage of already established fake news detection validators in our data collection to improve our final model moving forward.

# Table of Contents

[Acknowledgments 2](#_Toc140607749)

[ABSTRACT 3](#_Toc140607750)

[Table of Contents 4](#_Toc140607751)

[List of Figures 6](#_Toc140607752)

[CHAPTER 1. Introduction 7](#_Toc140607753)

[1.1 - Motivation 8](#_Toc140607754)

[1.2 - Aim and Objectives 8](#_Toc140607755)

[1.3 - Technologies 9](#_Toc140607756)

[1.3.1 - Fields of Study 9](#_Toc140607757)

[NLP 9](#_Toc140607758)

[Graph theory 10](#_Toc140607759)

[Machine Learning 10](#_Toc140607760)

[1.3.2 - Technical Fields 11](#_Toc140607761)

[Python 11](#_Toc140607762)

[HTML, CSS, JS 11](#_Toc140607763)

[1.4 - Challenges 12](#_Toc140607764)

[CHAPTER 2. Background & Literature Review 13](#_Toc140607765)

[2.1 - Background 13](#_Toc140607766)

[2.2 - Literature Review 14](#_Toc140607767)

[2.2.1 - Projects 14](#_Toc140607768)

[2.2.2 - Technologies studied 16](#_Toc140607769)

[Cosine similarity 19](#_Toc140607770)

[Neo4j 21](#_Toc140607771)

[**Dgraph** 22](#_Toc140607772)

[PyTorch 22](#_Toc140607773)

[Graph classification neural network 24](#_Toc140607774)

[Graph Attention Networks (GATs): 25](#_Toc140607775)

[CHAPTER 3. System Development 26](#_Toc140607776)

[3.1 – Requirements 26](#_Toc140607777)

[3.1.1 - Product Description 26](#_Toc140607778)

[3.1.2 - System Objectives 26](#_Toc140607779)

[3.1.3 - Operating Environments 26](#_Toc140607780)

[3.1.4 - Functional Requirements 26](#_Toc140607781)

[3.1.5 - Non-Functional Requirements 26](#_Toc140607782)

[3.1.6 - User Requirements 28](#_Toc140607783)

[3.1.6 - System Requirements 28](#_Toc140607784)

[3.2 - System Analysis 29](#_Toc140607785)

[3.2.1 - Activity Diagram 29](#_Toc140607786)

[3.2.2 - Class Diagram 30](#_Toc140607787)

[3.2.3 - Use Case Diagram 31](#_Toc140607788)

[3.2.4 - Sequence Diagram 32](#_Toc140607789)

[3.2.5 - System Architecture 33](#_Toc140607790)

[3.2.6 - Deployment Diagram 34](#_Toc140607791)

[3.3 - Design Sketch 35](#_Toc140607792)

[3.3.1 - Desktop 35](#_Toc140607793)

[3.3.2 - Mobile 36](#_Toc140607794)

[CHAPTER 4. Implementation Process 37](#_Toc140607795)

[4.1 - Data sources 37](#_Toc140607796)

[4.1.1 - True Dataset sources 37](#_Toc140607797)

[4.1.2 Fake Dataset sources 38](#_Toc140607798)

[4.2 - Data collection 38](#_Toc140607799)

[4.3 - Data cleaning 39](#_Toc140607800)

[4.4 - Summery extraction 41](#_Toc140607801)

[4.5 – Understanding the structure of our directed nodes 42](#_Toc140607802)

[4.7 – Transforming the string data into Vectors 42](#_Toc140607803)

[4.8 – Transforming the Vector data into classes 43](#_Toc140607804)

[4.9 - Applying the Graphical Probability model (Bayesian Networks) 43](#_Toc140607805)

[CHAPTER 5. Future Work 47](#_Toc140607806)

[CHAPTER 6. Conclusion 48](#_Toc140607807)

[References: 49](#_Toc140607808)

[Turn It In Results: 50](#_Toc140607809)

# List of Figures

[Figure 1 - Aim img 8](#_heading=h.184mhaj)

[Figure 2 - NLP Visual](about:blank) 9

[Figure 3 - Graph Visual](about:blank) 9

[Figure 4 - Machine Learning Types 10](#_heading=h.45jfvxd)

[Figure 5 - Stance Detection Visual](about:blank) 13

[Figure 6 - Textual Analysis Layers 14](#_heading=h.3bj1y38)

[Figure 7 - Neuron Visual](about:blank) 15

[Figure 8 - Neural Network Visual 15](#_heading=h.1qoc8b1)

[Figure 9 - Opinosis Visual](about:blank) 16

[Figure 10 - TextRank Steps 17](#_heading=h.3oy7u29)

[Figure 11 - Naive Bayes Classification](about:blank) 19

[Figure 12 - Cosine Similarity visual](about:blank) 19

[Figure 13 - Cosine Similarity Visual2 19](#_heading=h.338fx5o)

[Figure 14 - Word2Vec Visual 20](#_heading=h.42ddq1a)

[Figure 15 - Word2Vec Visual2](about:blank) 20

[Figure 16 - CBOW vs Skip-gram 20](#_heading=h.2hio093)

[Figure 17 - Dgraph Visual](about:blank) 21

[Figure 18 - GNN Visual 23](#_heading=h.2uxtw84)

[Figure 19 - GAT Visual](about:blank) 23

[Figure 20 - Activity Diagram 26](#_heading=h.3ls5o66)

[Figure 21 - Class Diagram 27](#_heading=h.20xfydz)

[Figure 22 – Use Case Diagram 28](#_heading=h.4kx3h1s)

[Figure 23 - Sequence Diagram 29](#_heading=h.302dr9l)

[Figure 24 - Sequence Architecture 30](#_heading=h.3z7bk57)

[Figure 25 - Deployment Diagram 31](#_heading=h.thw4kt)

[Figure 26 - Desktop Sketch1 32](#_heading=h.3dhjn8m)

[Figure 27 - Desktop Sketch2 32](#_heading=h.1smtxgf)

[Figure 28 - Mobile Sketch1 33](#_heading=h.4cmhg48)

[Figure 29 - Mobile Sketch2 33](#_heading=h.2rrrqc1)

[Figure 30 - Web Scraping Img 36](#_heading=h.356xmb2)

[Figure 31 - Node Example Img](about:blank) 39

[Figure 32 - Graph classification visual 42](#_heading=h.3im3ia3)

[Figure 33 - Project Steps](about:blank) 44

# CHAPTER 1. Introduction

The world is transforming quickly. There are undoubtedly many benefits to living in a digital age, but there are also drawbacks. Different problems exist in this digital age, and fake news is one of them. Fake news are simple to distribute, Spreading them can damage someone's or an organization's reputation. Propaganda against a person, group, political party, or organization is one possibility. There are numerous online platforms where bogus news can be disseminated. This covers Twitter, Facebook, and other sites.

Artificial intelligence includes machine learning, and techniques like text extraction and graph theory all assist in creating computers that can learn and carry out various tasks. Various machine learning algorithms, such as supervised, unsupervised, and reinforcement learning algorithms, are accessible. A data collection called the training dataset is used to first train the algorithms. It contains subjects, claims, entities, sources, and other information that was obtained using a text extraction method. These algorithms can be utilized to carry out many tasks after training. To carry out various tasks, we make use of machine learning, graph theory, and some statistical equations. These algorithmic techniques are frequently used to make predictions or find something that is concealed.

Users benefit from online platforms since they can simply get news. The issue is that this offers online criminals the chance to use these sites to distribute false information. An individual or society may be harmed by this news. Readers read the news and begin to believe it before it has been verified. Finding fake news is difficult since it is not a simple task. If fake news is not exposed quickly, it can spread, and eventually, everyone will start to believe it. Fake news can have an impact on people, groups, or political parties. The 2016 US election's fake news has had an impact on people's opinions and choices.

The detection of fake news is a topic of research for numerous academics. In this aspect, machine learning is proving to be beneficial. Various algorithms are being used by researchers to identify bogus news. The detection of fake news, according to researchers (Wang, 2017), is a major difficulty. For the purpose of identifying bogus news, they applied machine learning. Researchers (Zhou et al., 2019) discovered that the prevalence of fake news is rising over time. Because of this, it's important to spot phony news. To achieve this goal, numerous alternative algorithms are trained. Once taught, these algorithms will automatically recognize phony news.

The many research issues will be addressed by this examination of the literature. This literature study will demonstrate the value of machine learning, graph theory, and statistical equations in identifying bogus news. The use of these algorithms for spotting fake news will also be covered. In the literature review, false news detection algorithms will be covered.

The other sections of the essay are organized as follows: methodology, showing the research topics, model of the search strategy used for this literature review, results and discussion, and conclusion. References are provided at the end for the papers covered in this literature study.

## 

## 1.1 - Motivation

The spread of social media and chat messaging apps has forever changed how people consume news. An estimated 66% of people in the middle east use social media to look for news daily.

And after the COVID-19 virus, there was a major decentralization of news sources from the newsroom to the hands of every citizen has increased rapidly, there are many advantages such as diversity of platforms, but it also has its drawbacks, from misinformation to manipulating public thoughts, fake news produces a negative impact on individuals, companies, and governments.

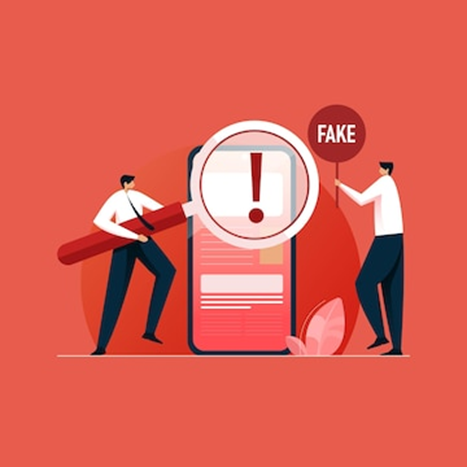
Fake News Detection is aimed to help citizens with a tool to at least get an indicator if the news they just observed was generally used in fake news or not and a full report of how it reached that conclusion.

The existing works for fake news detection in the Arabic language are very scarce, and Arabic datasets are not sufficient, but there have emerged some Arabic fake news validators on Facebook such as (Da Begad), (Akeed), (Kashif), and (Tahaqaq), these pages manually research and confirm the truth behind fake news stories. Our project will take advantage of this strong community of validators by using web scraping tools to generate an Open domain dataset. With this dataset, we can improve our built Bayesian Network model with new data every new post these validators provide.

## 

## 1.2 - Aim and Objectives

The purpose of this project is to provide people with a reliable and scalable solution to protect themselves from the rapid increase of misinformation and fake news, although this project won’t definitively classify an article as fake or true, it will provide an accuracy indicator to determine that article’s reliability. We aim to have an easy-to-use web page where the user would enter the URL or parts of the article and after a few seconds, the site would provide an accurate indicator as well as a full diagnosis of that article, consisting of (the trustworthiness of the source and author, what summaries the algorithm concluded were fake or true) all based on the data we collected. Our future objective is to implement this project in the Arabic language. We plan to use HTML for the front end and larvel for the back end.



*Figure 1 - Aim img*

## 1.3 - Technologies

### 1.3.1 - Fields of Study

To understand the complexity of the articles and describe them in a way understandable by the machine we would require the following fields.

#### NLP

*Figure - NLP visual*

*Figure - NLP Visual*

A branch of artificial intelligence called "Natural Language Processing" (NLP) focuses on how computers and human language interact. NLP aims to make it possible for computers to meaningfully comprehend, interpret, and produce human language. NLP encompasses activities including part-of-speech tagging, parsing, sentiment analysis, named entity recognition, and text categorization. Machine learning algorithms are the foundation of NLP approaches, which are taught on massive text corpora to discover linguistic patterns. These algorithms are used to create NLP models that are capable of carrying out a range of NLP activities. NLP is essential for helping computers comprehend human language and has a wide range of uses, including text-to-speech systems, social media monitoring, and customer support.

#### Graph theory

*Figure - Graph visual*

*Figure - Graph Visual*

Graphs are mathematical structures used to represent objects or entities and the relations between them, which makes them an excellent tool for NLP models, with a good enough dataset they can create a complex relational structure between words, dialects, and even hidden patterns that would go unnoticed without it.   
Graphs are essentially a set of nodes and edges that connect pairs of nodes. Graphs have the unique properties of better representation and clarity of data, informative data recall, and scalability.

#### Machine Learning

Machine learning is a subfield of artificial intelligence that focuses on the development of algorithms and statistical models that allow computers to improve their performance on a specific task over time through experience. Machine learning algorithms are designed to automatically learn patterns in data and make predictions or decisions based on that data.

*Figure 4 - Machine Learning Types*

There are three main types of machine learning: **supervised learning**, in which the algorithm is trained on a labeled dataset and makes predictions based on that training, **unsupervised learning**, in which the algorithm works with an unlabeled dataset to identify patterns or relationships in the data, and **reinforcement learning**, which focuses on how an agent should take actions in an environment to maximize a reward signal which the machine uses to improve its behavior over time. Machine learning has numerous applications across a wide range of industries, from healthcare and finance to retail and marketing, and is increasingly being used to solve complex problems and drive decision-making processes.[1]

Web Scraping

Web scraping is the practice of utilizing software to automatically scrape data from websites. It entails sending HTTP requests to the server of a website in order to download the HTML content of a web page, which is subsequently parsed to obtain the needed information. Prices, product descriptions, customer reviews, and a host of other pieces of information may all be extracted from websites using web scraping.

### 1.3.2 - Technical Fields

#### Python

Python is the greatest option since it has the most resources, support, and is the simplest to use when dealing with machine learning and natural language processing. The following libraries will be utilized in this project:

* **Newspaper3k** is a python library for extracting features of news articles such as title, author, text, and other metadata.[11]
* **TextRank** is a graph-based algorithm used to identify and extract the most important sentences or phrases in a piece of text, and to summarize the text by condensing it into a smaller number of sentences that capture its main content and meaning. We can use it to extract the claims of articles.[8]
* **pgmpy** is a machine learning python library that provides a set of tools for buildings and training Bayesian networks[21]
* **Larvel** is a light web framework for PHP that helps in building web applications.

#### HTML, CSS, JS

For developing the frontend of the site, we are using html, css, and js, as they are the standard for any website.

****

## 1.4 - Challenges

The biggest challenge is how we can build or use a model complex enough to represent the article’s intertwined structures. In this study, we are using the Naive Bayes graph model, but we are looking to expand to use a more complex graph probability model such as **Bayesian Networks**, or **Graph Attention Networks (GATs)**.

# CHAPTER 2. Background & Literature Review

## 2.1 - Background

*Figure - Stance detection visual*

*Figure - Stance Detection Visual*

Many projects tried to tackle the problem of the fake news epidemic, but the problem seemed too large to be solved by humans and too complex to be solved by machines, there was even a challenge in 2017 called FNC (fake news challenge), where they mainly tried to solve the problem using stance detection techniques. Stance detection can be used to determine the credibility of a news article by analyzing its position or attitude toward a specific event or topic. This involves classifying the article as being supportive of the event, neutral, or opposing, and comparing it to other articles on the same topic to identify inconsistencies or conflicting viewpoints. By detecting the stance of an article, it can be determined if it is presenting a balanced or biased perspective and if it aligns with other credible sources, which can be an indicator of the reliability and accuracy of the news article. Stance detection is one of the many techniques used to detect fake news.[1]

We noticed an extreme lack of projects trying to solve it using the Arabic language. Our goal in this project is to change that, by taking inspiration from these other projects as well as the NLP algorithms to create a fake news detection project that supports the Arabic language.

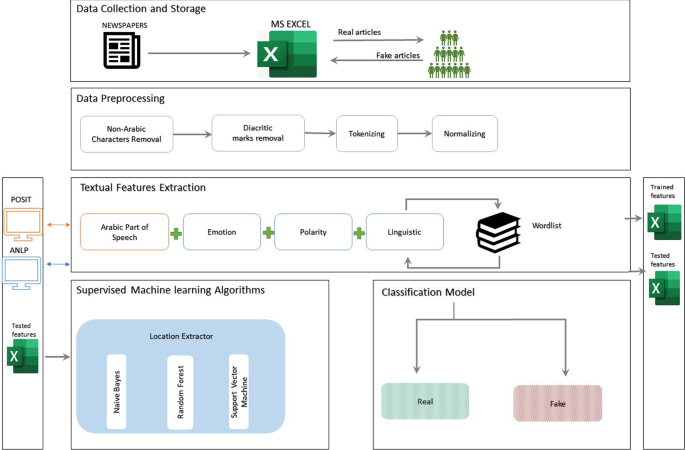
## 2.2 - Literature Review

### 2.2.1 - Projects

Arabic Fake News Detection Based On Textual Analysis:

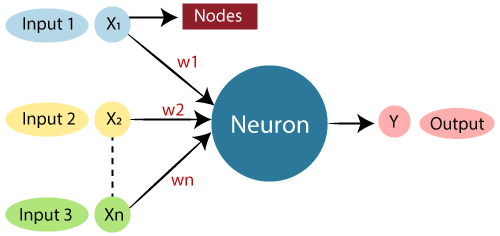
This study is one of the few that try to solve this problem using the Arabic language, and due to the lack of resources for Arabic datasets, they collected a Real News Dataset using a python scraper on the topic of Hajj. As for the Fake News Dataset, they collected it using crowdsourcing.

After Preprocessing and cleaning the data, they solve the problem by introducing a supervised machine learning model that classifies Arabic news articles based on their context’s credibility.[7]



*Figure 6 - Textual Analysis Layers*

Using Artificial Neural Network To Identify COVID-19 Misinformation:

This study chooses to tackle the COVID-19 Misinformation problem with NLP neural network models. The models that were tested were K-Nearest Neighbor (k-NN), Naïve Bayes, Support Vector Machines (SVMs), but the model that gave the best results was the Artificial Neural Networks model (ANNs).  
ANNs are powerful classification techniques made up of a network of connected neurons in various layers. A Neural Network consists of an input layer, hidden (or more) layers), and an output layer. The data is passed to nodes (i.e., neurons) from one layer to the other and each node contains an activation function that represents a mathematical transformation of data. 

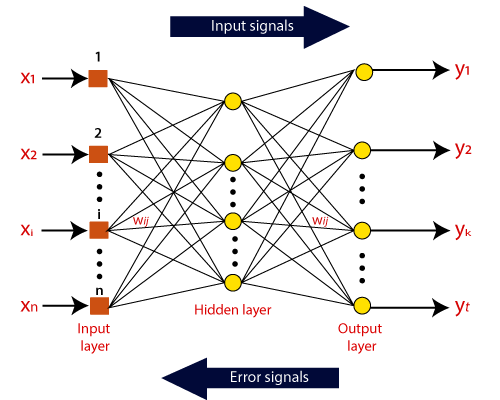
*Figure : Neuron*

*Figure - Neuron visual*

*Figure - Neuron Visual*

The weight and bias adjustments to the inputs determine whether or not to activate the neuron. This algorithm is applied to complex and uncorrelated datasets.

ANNs can work with numerical data. So before using ANN, problems must be converted into numerical values. The rendering mechanism to be solved here will directly affect network performance. Depends on the capabilities of the user.[2]



*Figure 8 - Neural Network Visual*

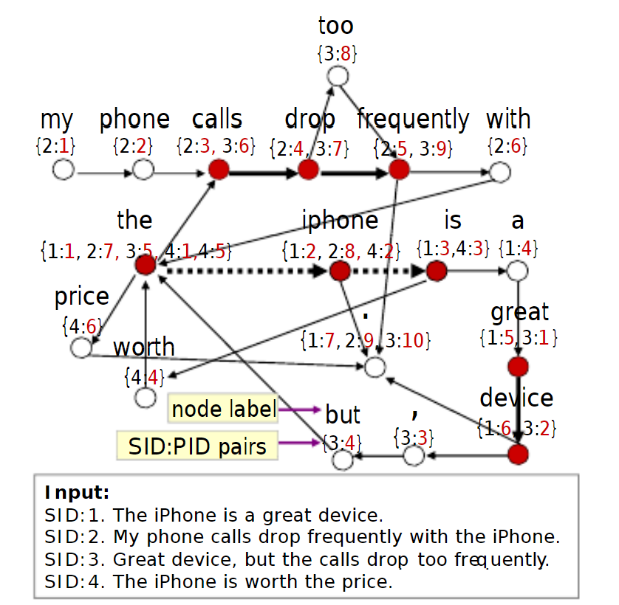
An intelligent cybersecurity system for detecting fake news on social media websites.

The proposed system is based on verifying the accuracy of the news published by news websites, this project doesn’t use machine learning to classify news like the other projects, instead, it works on two factors. First the similarity of news text across fake news datasets such as Kaggle; second, the website’s ranking. The ranking system uses data from two sources, such as RankAPI and Alexa. They gave 50% of the score to text similarity and 50% for news site rating, and then calculate the news accuracy by adding these percentages together.[9]

### 2.2.2 - Technologies studied

There are two types of summarizations (Extractive summarization, and Abstractive summarization), we determined that extractive summaries are more suitable for our project. And for extracting summaries we found Opinosis and Textrank.

Opinosis: A Graph-Based Approach to Abstractive Summarization of Highly Redundant Opinions:

Opinosis is a graph-based summarizing method for locating and highlighting key words and phrases in a document. It is intended to provide concise, non-redundant summaries that accurately reflect the important points and viewpoints of the original text. Three main steps make up the summarizing process for opinosis:[6]

*Figure - Opinosis graph*

*Figure - Opinosis Visual*

**-Phrase Extraction:** The first step is to extract candidate phrases from the text. These phrases are then used as the nodes for the graph that will be constructed in the next step.

**-Graph Construction:** The phrases are then linked together to create a graph, where each extracted phrase is represented by a node and links between phrases are represented by edges. The edges may be based on co-occurrence, similarity, or dependence links, among other sorts of linkages.

**-Graph Traversal:** The last step is to navigate the graph and pick out the key words and phrases. This is usually accomplished by using a graph-based algorithm, like PageRank or HITS, which rates each node in the graph according to its significance. The summary is produced using the terms that received the highest ratings since they are thought to be the most crucial.

Opinosis is especially helpful for summarizing lengthy and complex texts, such as news articles, research papers, and customer evaluations, where it may pinpoint the main viewpoints and moods conveyed in the text.

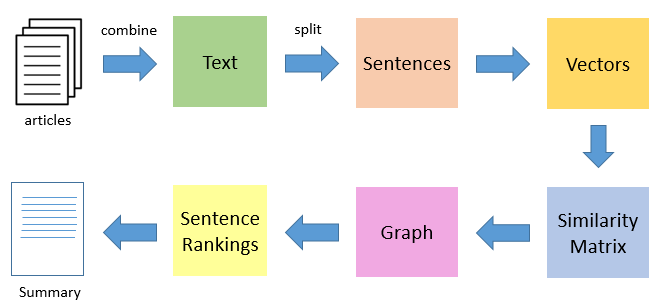
TextRank: A Graph-Based Ranking Model for Text Processing

TextRank is a graph-based algorithm that works by representing the text as a graph, with each sentence as a node, and the edges between them representing the similarity or relevance of the sentences to one another. The algorithm starts by creating a similarity matrix between all the sentences, where the similarity score is calculated using a similarity measure such as cosine similarity.[8]

Next, the algorithm constructs a graph by connecting each sentence to the most similar sentences, creating edges between them. The edges are assigned weights based on the similarity scores.

After the graph is constructed, the algorithm applies PageRank to the graph, which is an algorithm for ranking nodes in a graph. The PageRank algorithm assigns a score to each sentence based on the number and weight of the edges connecting it to other sentences. The sentences with the highest scores are considered the most important and are used for summarization and keyword extraction.

The whole process is iterative, and the scores are recalculated in each iteration until they converge. The final scores of the sentences are the output of the TextRank algorithm, and the higher the score, the more important the sentence is considered to be.



*Figure 10 - TextRank Steps*

TextRank can also be modified to consider different similarity measures, using different graph representations such as using words as nodes.

In this study we would experiment with using both sentences and words as nodes, to find the best accuracy.

While both methods use text similarity and graph theory they are different in how they implement it. But TextRank is considered to be more efficient and accurate than opinosis, as opinosis is mostly used to summarize a large number of opinions on a comment section or feedback page.

Some of the **advantages** of TextRank include:

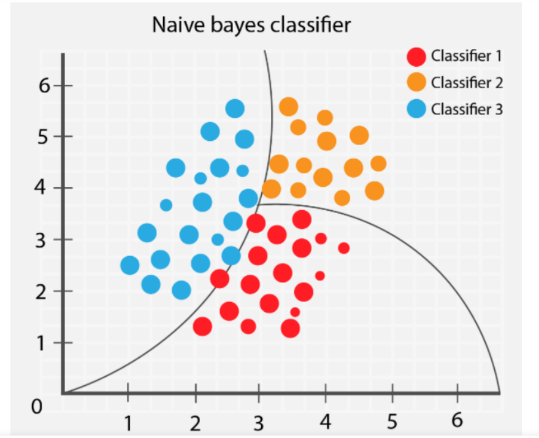
1. It is unsupervised, meaning it does not require labeled data to train, making it easy to use.
2. It can be applied to any type of text, including long documents and articles.
3. It produces high-quality summaries and extracts meaningful keywords.

Some of the **disadvantages** of TextRank include:

1. It can be computationally expensive, especially for large texts.
2. It may not always produce the most accurate summaries, as it is based on the graph-based ranking algorithm.
3. It can be sensitive to the quality of input text.

Overall TextRank is a good choice for summarization or keyword extraction but it can be improved with other techniques like using a **Named Entity Recognition tool** or using a **supervised approach**.

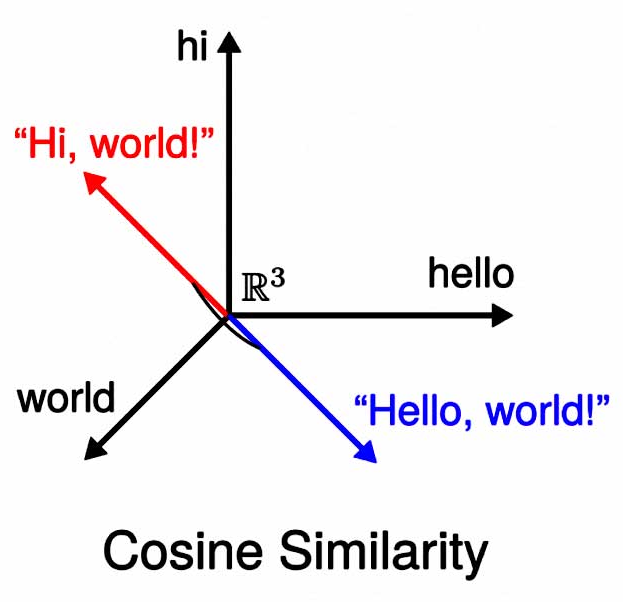
Naive Bayes:

The Naive Bayes classifiers are a collection of classification algorithms based on Bayes’ Theorem. It works with probabilistic ML algorithms and assumes that all features are equal and independent of one another. It calculates the likelihood of the target word being placed in the Fake or True dataset, as a ratio. This method only gives high accuracy if the user understands the nature of their data and the features in your dataset are independent as well as having no relation to each other. 

*Figure - Naive Bayes visual*

*Figure - Naive Bayes Classification*

#### Cosine similarity

The cosine of the angle between any two non-zero vectors in an inner product space is used to measure how similar the vectors are. In other words, it computes the cosine of the angle between two vectors to determine how similar they are.

*Figure - Cosine Similarity visual*

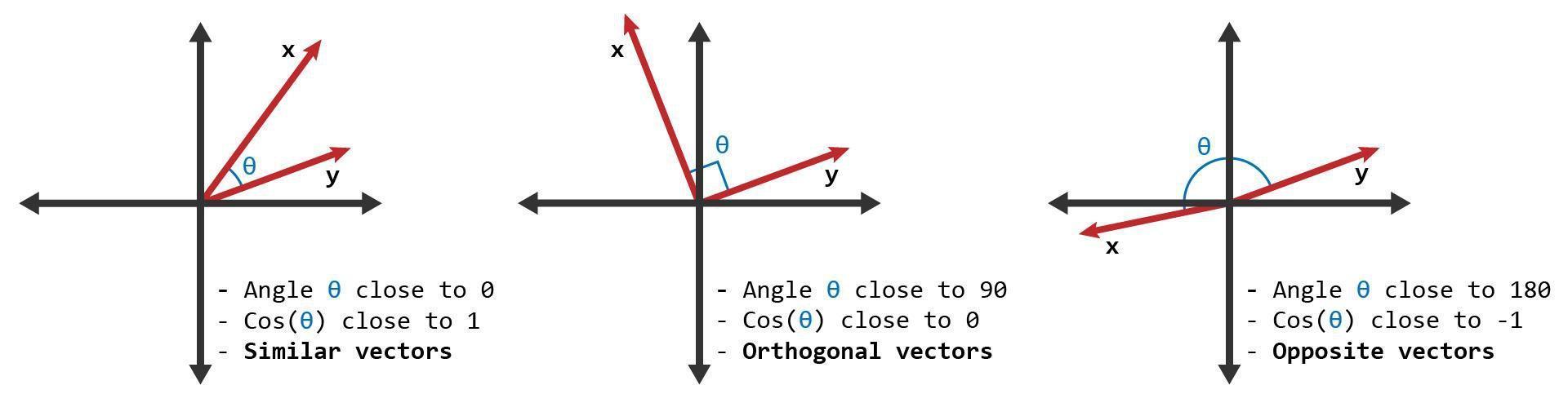
*Figure - Cosine Similarity visual*

The similarity between two sentences in a text is gauged using cosine similarity in the context of text summarization. By encoding each sentence as a bag-of-words (BOW) representation, the vectors for each sentence are constructed in order to calculate the cosine similarity. Each distinct word in the phrase is regarded as a vector dimension in this representation, and each dimension's value is the number of times that word appears in the sentence.

The dot product of the vectors divided by the product of the vector magnitudes is used to calculate the cosine similarity between two vectors A and B.

The cosine similarity scales from -1 to 1, with 1 denoting that the two vectors are identical, 0 denoting that they are perpendicular to one another, and -1 denoting that they are completely at odds with one another.

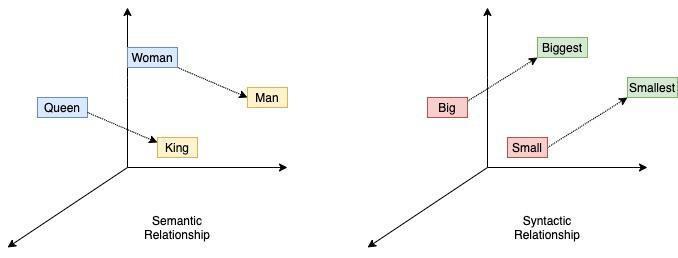
Because it is simple to compute and is unaffected by the magnitude of the vectors, only by their direction, cosine similarity is a frequently used similarity measure in text processing.



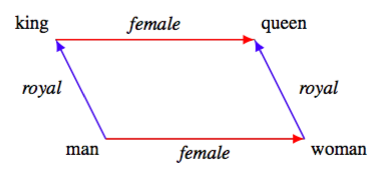
*Figure 13 - Cosine Similarity Visual2*

Word2Vec & Softmax:

Word2Vec is an NLP algorithm used to transform text to a vector representation that gives meaning to a machine, these vectors give meaning and relationships to the words such as gender.[19]



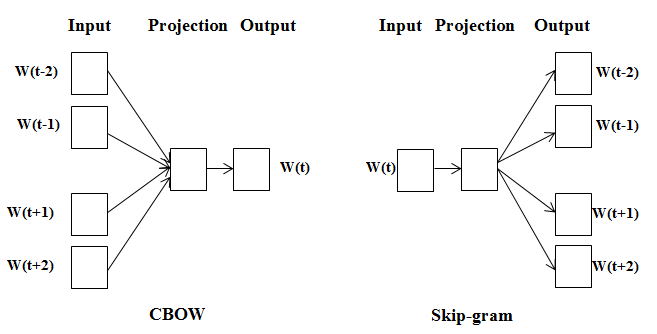
*Figure 14 - Word2Vec Visual*

, and we can then apply mathematical operations on those vector numbers to manipulate and form new vectors as needed. A popular example is the   
(King – Man + Woman = Queen)

*Figure - Word2Vec visual2*

*Figure - Word2Vec Visual2*

The two most popular models are Continuous bag of words(CBOW) and Skip-gram. CBOW predicts a target word from a list of context words, whereas a Skip-gram does the opposite, it predicts the words before and after the current chosen word.



*Figure 16 - CBOW vs Skip-gram*

The final layer of both CBOW and Skip-gram is Softmax. Softmax is a mathematical function used to convert the vector results from Word2Vec into probabilities and then divide those probabilities into categories, it does this by applying the probability equation to each value.[4]

**probability = exp(value) / (exp(value1) + exp(value2) + exp(value3))**

GloVe: Global Vectors for Word Representation

Glove, short for Global Vectors for Word Representation, is an unsupervised learning algorithm for generating vector embedding much like word2vec, but it shines in its ability to capture both the syntactic and semantic relationships between words. [22]

The implementation of Glove in pgmpy expands the library’s capabilities by allowing researchers to leverage the word embeddings within their probabilistic graphical models.

**For storing our graph data we found the best options would be Neo4j and Dgraph**

#### Neo4j

**Neo4j** is a native, open-source graph database management system. It is designed to handle massive amounts of graph data. Neo4j uses a graph-based storage format for data, enabling highly connected data and quick, effective searching.

One of Neo4j's standout features is the Cypher query language, a declarative language for querying graph data that is inspired by SQL. Cypher makes graph data querying simple and intuitive, making it straightforward to draw conclusions and relationships from the data.

Additionally, Neo4j enables ACID transactions, which guarantee data isolation and consistency. Neo4j also has horizontal scalability, which enables it to be distributed across numerous machines to manage massive amounts of data and heavy traffic.

Neo4j is a great tool for working with graph data because of its wealth of capabilities, which include support for indexing, full-text search, geospatial data, and real-time data processing.

Neo4j is a good fit for use cases like master data management, fraud detection, recommendation systems, and social networks. Additionally, it is utilized in a number of sectors, including e-commerce, healthcare, and banking.

Neo4j's drawback is that Dgraph is entirely free, but Neo4j isn't.

#### **Dgraph**

*Figure - Dgraph Visual*

**Dgraph** is a distributed, native graph database that is open-source and created using the Go programming language. It is built to handle massive amounts of graph data and is tuned for fast, low-latency queries.

Dgraph has horizontal scalability, allowing it to handle enormous traffic and data volumes by being distributed over numerous workstations. It is suitable for use cases like real-time analytics, recommendation systems, and social networks since it offers real-time data processing and low-latency queries.

Dgraph is extremely powerful in use situations where these features are crucial because it also comes with built-in support for full-text search, geolocation, and faceting.[1]

Dgraph is an effective tool for managing massive amounts of graph data, with an emphasis on performance and scalability.

*Figure - Dgraph visual*

**As for graph machine learning we can use PyTorch**

#### PyTorch

PyTorch is an open-source machine-learning library for Python that provides a set of tools for building and training neural networks. It was created by Facebook's AI Research team, and PyTorch is currently responsible for its upkeep.

In comparison to other deep learning frameworks like TensorFlow, PyTorch's dynamic computational graph offers greater flexibility and ease of use. This implies that PyTorch will automatically compute gradients over the graph even if the user changes the graph in real time while the program is running. Debugging and experimentation may become simpler and quicker as a result.

Additionally, PyTorch offers a large selection of pre-built neural network modules and layers, including linear, convolutional, and recurrent layers. Several pre-processing and data-loading methods are also included, which make it simple to load and prepare data for use in training and assessing models.

PyTorch also has a growing ecosystem of tools and libraries that make it more powerful and versatile. For computer vision, audio processing, natural language processing, meta-learning, and other applications, there are libraries like torchvision, torchaudio, torchtext, torchmeta, and torchfusion. Additionally, it is simple to locate tutorials, resources, and pre-trained models thanks to PyTorch's robust community and widespread use in both industry and academia.

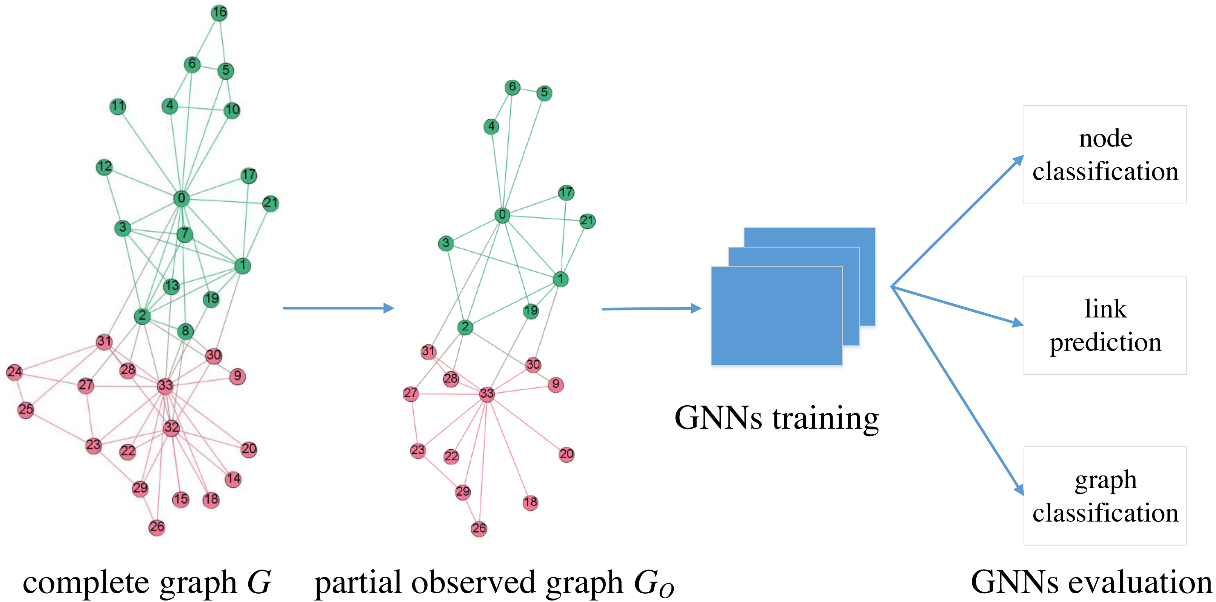
Now we are interested in the library Blitz which is a simple and extensible library to create Bayesian Neural Network layers on the top of PyTorch..[15]

#### Graph classification neural network

A Graph Neural Network (GNN) is a type of neural network that is designed to operate on graph-structured data. Graphs are a powerful way to represent many types of data, such as social networks, molecular structures, and transportation networks. GNNs are designed to learn patterns and representations from graph data by propagating information through the graph's edges and nodes.

GNNs are typically composed of multiple layers, where each layer performs a graph convolution operation. The convolution operation is a variation of the traditional convolution operation used in image processing, which is adapted to the graph structure. The convolution operation allows GNNs to aggregate information from the neighborhood of each node in the graph.

There are different types of GNNs, such as Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and Graph Recurrent Networks (GRNs). Each of these architectures has its own strengths and weaknesses, and the choice of which to use depends on the specific problem and dataset at hand.



*Figure 18 - GNN Visual*

All three types have the ability to perform tasks such as node classification, link prediction, and more, but we found that for our fake news classification problem, GAT is more suitable.

#### Graph Attention Networks (GATs):Graph Attention Networks Under the Hood | by Giuseppe Futia | Towards Data Science

*Figure - GATs visual*

A specific type of GNN called a GAT uses an attention mechanism to determine the importance of the nodes that are close to each node. GATs can focus their attention on important nodes while ignoring less important ones thanks to the attention mechanism. GATs have been used for a variety of tasks, such as node classification, connection prediction, and graph classification.[20]

*Figure - GAT Visual*

# 

# CHAPTER 3. System Development

## 3.1 – Requirements

### 3.1.1 - Product Description

Our product would be a web site application, that allows the user to check if their desired article is fake or true, by providing them with an option to insert either the article details like (title, author, source, text) or the URL of that article and out web scrapper would extract those details. After clicking check, an indicator would show the probability of the overall article being fake.

### 3.1.2 - System Objectives

We aim to solve the problem of fake news detection with a unique approach instead of just using a black box model such as BERT. And our approach is to use TextRank, and cosine simularity with Bayesian networks.

### 3.1.3 - Operating Environments

It should be as widely available as possible, so it is useable from any browser that can connect to the website.

### 3.1.4 - Functional Requirements

1. **URL Search**: Users should be allowed to submit news articles URL for fact-checking and the identification of false news.
2. **News verification**: Using a variety of fact-checking methods, the system should ascertain the reliability of the provided news stories.
3. **Display of verification results**: The system should show the verification results, indicating whether or not the news piece is authentic.
4. **Feedback**: Users should be able to report news articles they believe to be false.
5. **An administrative dashboard** should be accessible to administer the platform's content and keep track of activity.
6. The platform should have a responsive design that enables access from a variety of devices and screen sizes.

### 3.1.5 - Non-Functional Requirements

1. **Performance**: There should be less downtime and the platform should be quick to load all pages.
2. **Scalability**: The platform should be able to accommodate an expanding user base and volume of news articles without noticeably degrading its performance.
3. **Security**: To guard against unauthorized access, hacking, and data breaches, the platform should have the necessary security measures in place.
4. **Usability**: The platform ought to feature an easy-to-use interface that is also simple to traverse.
5. **Compatibility**: The platform should be compatible with a range of browsers and devices.
6. **Reliability**: The platform should function reliably, with few to no errors or flaws.
7. **Maintainability**: To make it easy to maintain and update, the platform should have a clear codebase and a well-documented architecture.
8. **Compliance**: The platform is required to follow all laws and regulations that are relevant, including those that cover data protection and fact-checking standards.

### 3.1.6 - User Requirements

1. **User-friendly interface**: The platform should have a simple, clear, and easy-to-use user interface.
2. **Results of the news verification process**: Users should have access to the findings and be able to view the sources that were used to conduct the news verification.
3. **User input**: Users should be able to comment on the verification findings to help the fake news detection algorithm become more accurate.
4. **News search**: Users should be able to enter either keywords and categories or URLs.

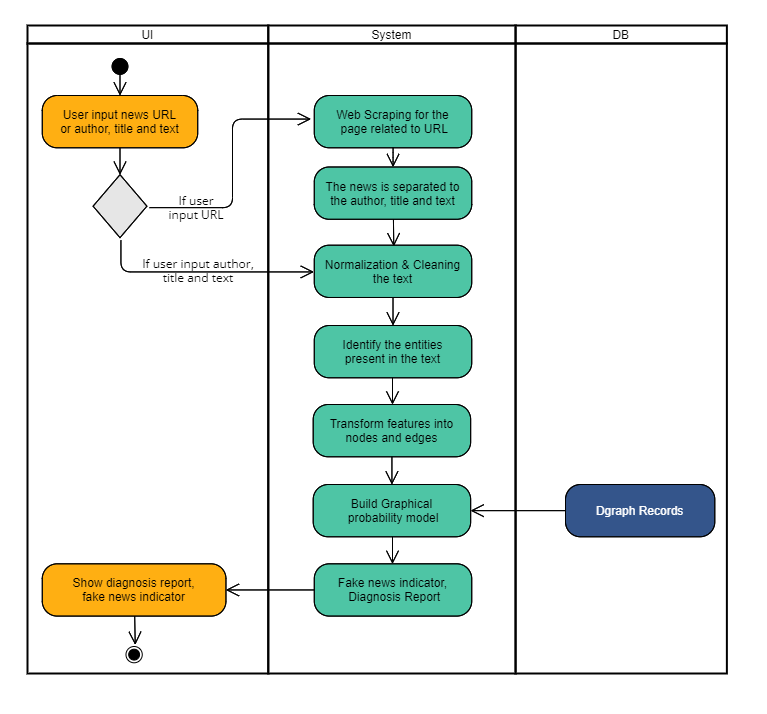
### 3.1.6 - System Requirements

1. **Operating system**: A common operating system, such as Windows, macOS, or Linux, should be compatible with the platform.
2. **Hosting**: The platform should be hosted on a web server, such as Apache or Nginx.
3. **Database management system**: User information and news articles should be stored in a database management system like GraphQL, Dgraph, MySQL or PostgreSQL.
4. **Programming languages**: PHP, Html, and python, should be utilized to create the platform.
5. **Front-end framework**: To create the user interface, a front-end framework like React, vuejs or Angular with HTML and CSS should be employed.
6. **Back-end framework**: The server-side logic should be created using a back-end framework, such as larvel.
7. **Storage**: Enough space should be made available for the user data and news articles to be stored.
8. **Bandwidth**: There should be enough bandwidth to accommodate several people accessing the platform at once.
9. **Security**: To guard against unauthorized access and data breaches, the proper security measures should be put in place, such as firewalls and SSL certificates.

## 3.2 - System Analysis

### 3.2.1 - Activity Diagram

Activity Diagram as shown in figure 20, shows how tasks are coordinated to deliver the service, which may be at various levels of abstraction.

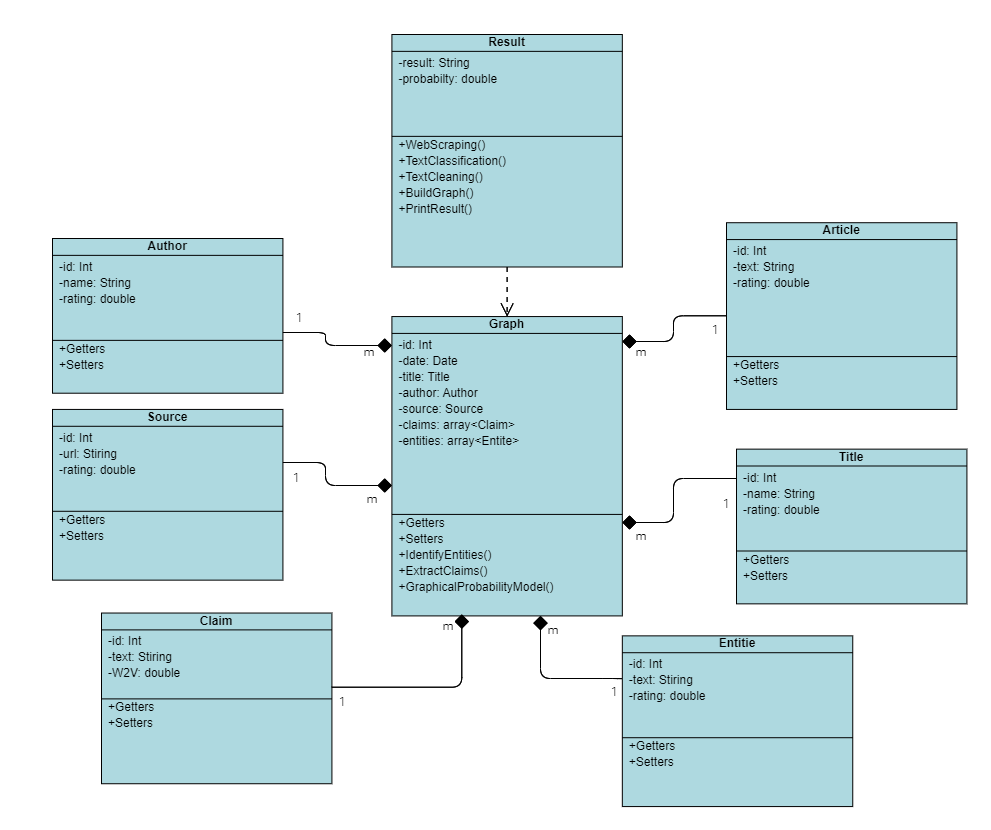


*Figure 20 - Activity Diagram*

### 

### 3.2.2 - Class Diagram

A class diagram is a particular kind of static structure diagram that illustrates a system's classes, attributes, operations, and relationships between objects, as shown in figure 21.

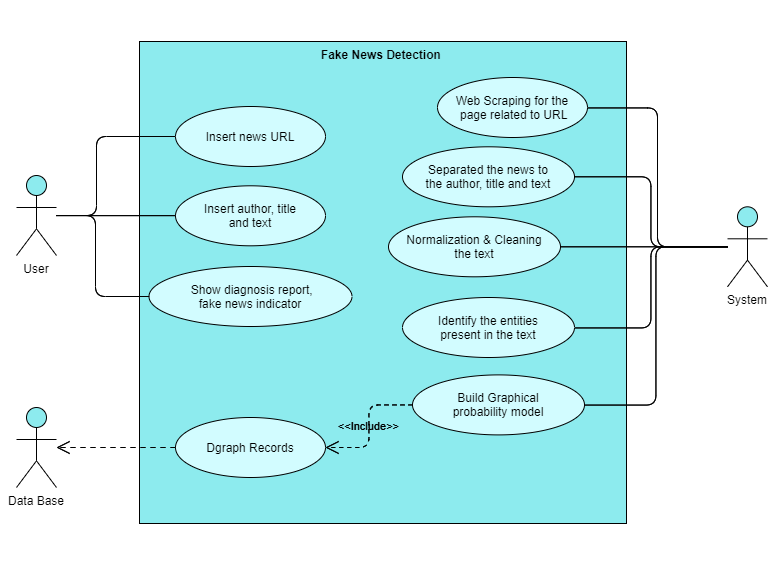


*Figure 21 - Class Diagram*

### 

### 3.2.3 - Use Case Diagram

A use case diagram is a visual representation of the interactions between users and a system, used to describe the system's functionality in terms of user goals and system responses, as shown in figure 22.

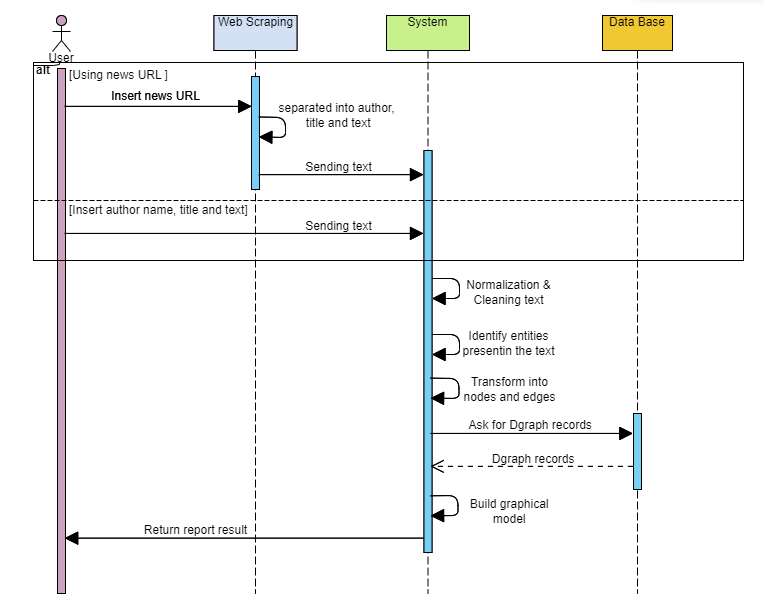


*Figure 22 – Use Case Diagram*

### 

### 3.2.4 - Sequence Diagram

Sequence diagram, as shown in figure 20, is a UML diagram that shows the interactions between objects or components in a system over time, illustrating the sequence of messages and their order.

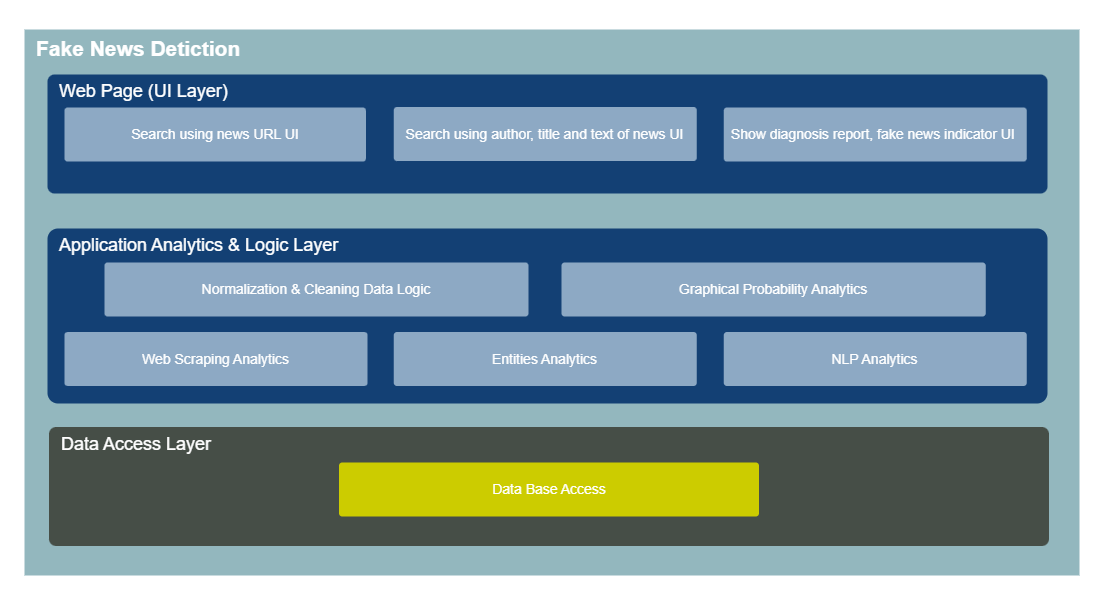


*Figure 23 - Sequence Diagram*

### 

### 3.2.5 - System Architecture

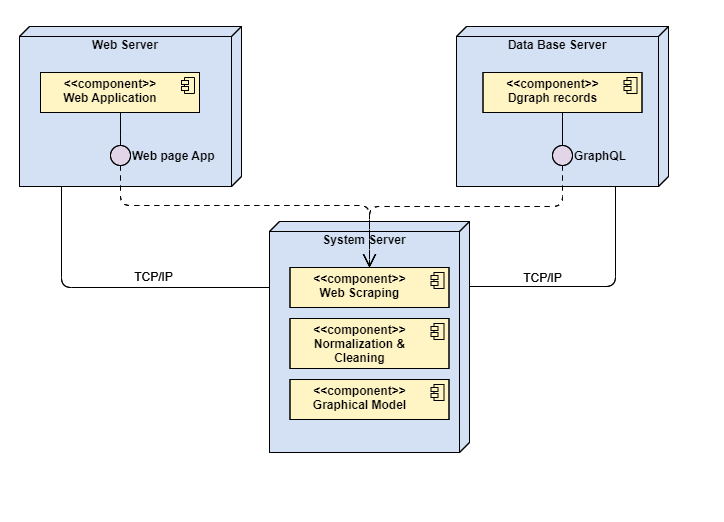
System architecture as shown in figure 24, refers to the organization and structure of a system's components, their relationships, and the principles governing their design, providing a high-level view of how they work together to achieve the system's goals.



*Figure 24 - Sequence Architecture*

### 3.2.6 - Deployment Diagram

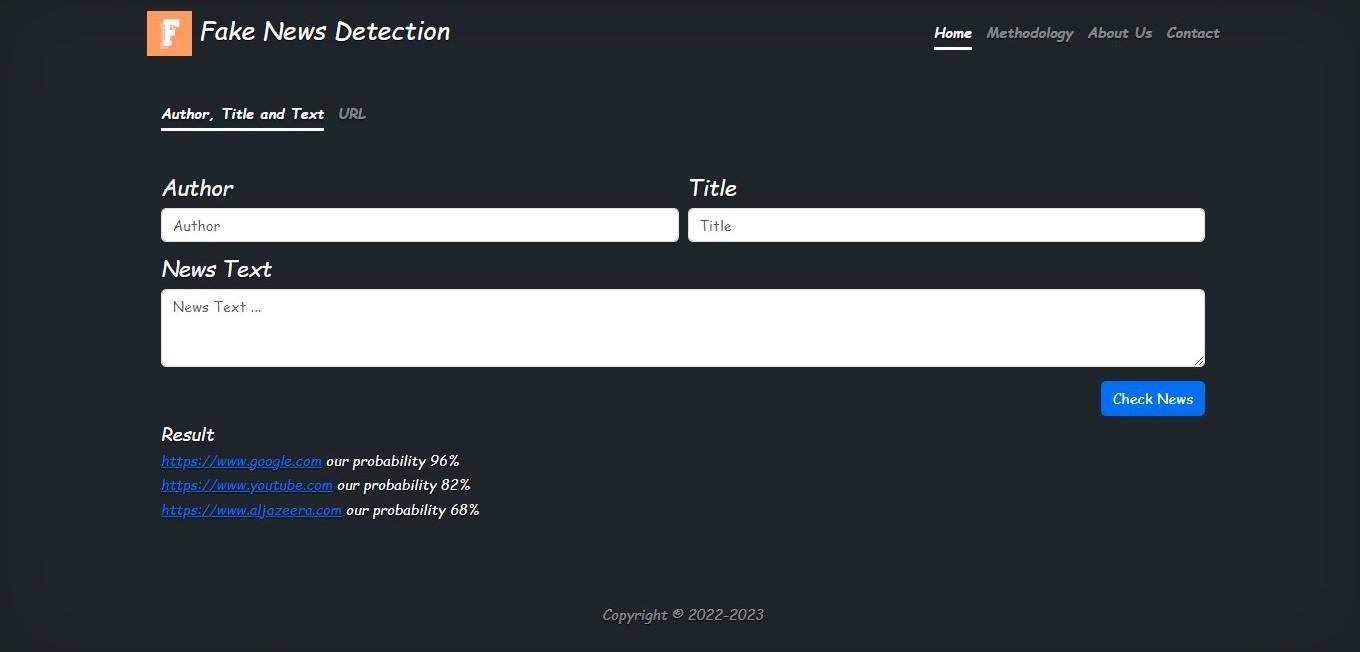
A deployment diagram is also a type of UML diagram that shows the physical deployment of software components to hardware devices and the relationships between them, providing a visualization of the system's deployment architecture, as shown in figure 25.



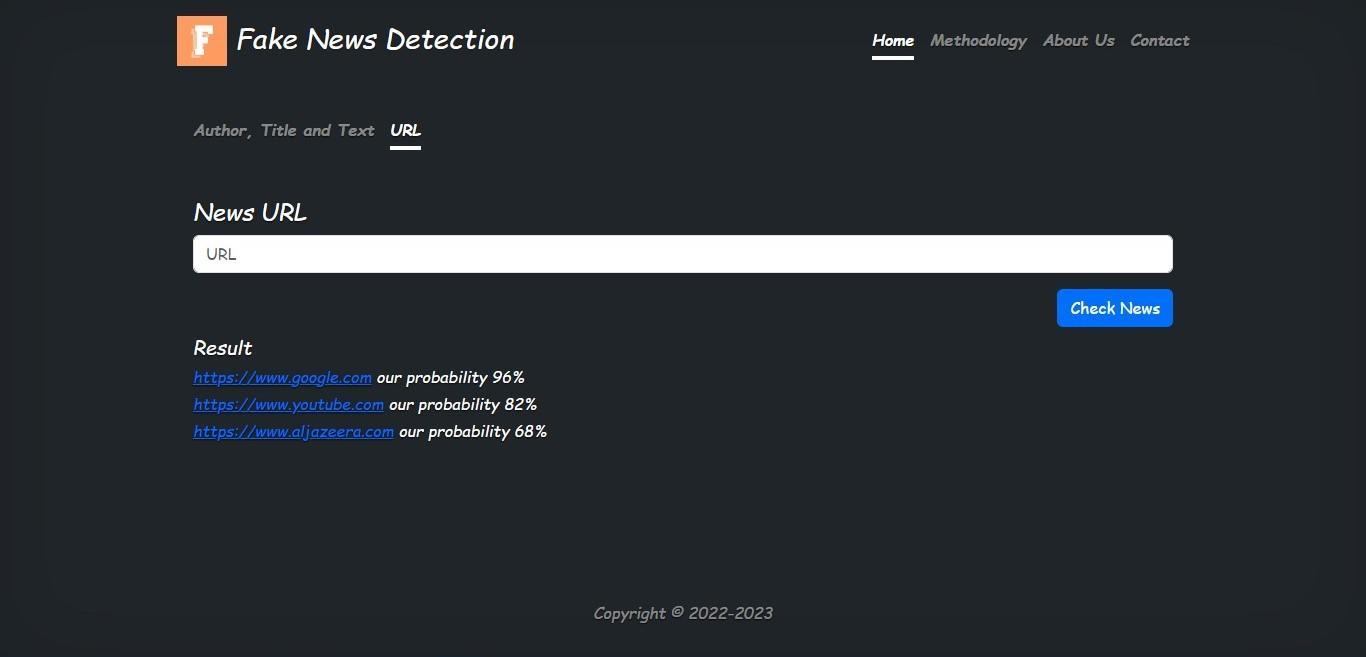
*Figure 25 - Deployment Diagram*

## 3.3 - Design Sketch

### 3.3.1 - Desktop

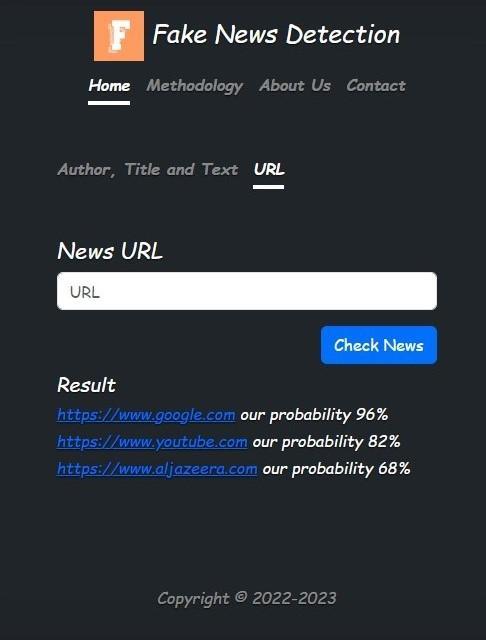
In the following two figures we can see two ways a user can interact with the system, either via features such as (author, title, text), or by URL link.  


*Figure 26 - Desktop Sketch1*



*Figure 27 - Desktop Sketch2*

### 3.3.2 - Mobile



*Figure 28 - Mobile Sketch1*



*Figure 29 - Mobile Sketch2*

# CHAPTER 4. Implementation Process

## 4.1 - Data sources

For our classification network to work we require two types of data, the first is the true dataset, and second is our fake data set.

### 4.1.1 - True Dataset sources

we can’t be one hundred percent sure that all posts by the news sources we collected are always true, so our project aims to provide the probability classification of the news article a user wants to check, of it being classified as fake or true based on out collected database network, as for our data itself, we have collected the following fifteen well-established, trusted news websites that we can consider to be reliable and accurate:

* The New York Times
* The Washington Post
* The Wall Street Journal
* BBC News
* CNN
* Reuters
* Associated Press (AP)
* NBC News
* ABC News
* CBS News
* Fox News
* NPR
* Al Jazeera
* BBC World Service
* ESPN

### 4.1.2 Fake Dataset sources

As we cant find explicitly fake articles, we can get the next best thing, Fake news revealing sites, we can use the fake news these sites are revealing and feeding it to our model as fake classification.

Here is a list of some popular ones:

* FactCheck.org
* PolitiFact
* Snopes
* The Washington Post Fact Checker
* FactChecking.org
* Associated Press Fact Check
* The Fact-Checking Project
* FactChecker.in
* Full Fact
* Reality Check by BBC News

As we move forward, we can also use Arabic fake news validators on Facebook, these pages manually research and confirm the truth behind fake news stories. Our project will take advantage of this strong community of validators so we can increase our Open domain database. Some of these pages are:

* (Da Begad)
* (Akeed)
* (Kashif)
* (Tahaqaq)

## 4.2 - Data collection

A Python module called Newspaper3k is used to retrieve and parse news articles from websites. It offers a thorough and versatile interface for working with news items from many sources and is made to be simple to use. The library combines machine learning algorithms and natural language processing techniques to extract pertinent data from articles, including the primary text, photos, authors, and publication dates. For journalists, data scientists, and other people wishing to collect and analyze news data, it is the perfect tool. [11]

Newspaper3k's capacity to automatically identify articles' core content, even when it is not explicitly stated, is one of its key strengths. A combination of methods, such as text extraction, keyword analysis, and pattern recognition, are used to achieve this. Articles published in a variety of languages, including English, German, French, Spanish, and more, can also be parsed using the library. As a result, it can be used as a flexible tool to work with news content from a variety of sources.

Newspaper3k offers a variety of tools for analyzing news article content in addition to extracting and parsing news articles. It can be used, for instance, to extract named items, do sentiment analysis, and produce statistics that summarize the articles. This makes it the perfect tool for reporting on a specific issue and for data-driven journalism.

Last but not least, Newspaper3k is an active open-source project that is maintained by a group of engineers. A broad user base and a well-documented library make it simple to get started using the library and locate support when necessary. Newspaper3k is unquestionably worth checking out if you're interested in working with news data and are seeking a simple-to-use but effective library.



*Figure 30 - Web Scraping Img*

## 4.3 - Data cleaning

In this section, current news text must pass in text preprocessing before searching for entities and claim process.

Text Pre-Processing:

Here, we must prepare the text with some processing that it become ready for the next process as the following steps show text processing(for the english language):

1. In the first step, we convert all letters in the text to lowercase.
2. This step removes all punctuations from the text and all non-ASCII characters.
3. Remove stopping words, such as “is”, “the”, “are”, “a”, “an”, etc., if they are not negative, but if followed by a negative like “isn’t” and “is not” we'll keep it in the text.
4. In this step, we are splitting all text into words, by using a space pattern for splitting it.
5. Finally, We return all text words to the stemme word, i.e., remove any suffix letters at the end of a word, for example, “Playing” ”played” and ”Plays” all of them refer to one basic word is ”Play” this is called word-stemming.

The text will be ready for the next stage.

This methodology is commonly used for text pre-processing because it helps to standardize the text data and remove irrelevant information, making it easier to perform NLP tasks such as fake news detection.

By converting text to lower case, removing special characters, and removing stop words, the text is reduced to its core meaning, reducing the dimensionality of the data and making it easier to analyze.

Returning all text words to the stemme word ensures that the data is accurate and it is free from errors, and standardizing the text helps to reduce variability in the data, making it easier to compare and analyze.

Overall, these steps help to prepare text data for NLP tasks in a way that is efficient, accurate, and effective, making this methodology a widely used best practice for text pre-processing in NLP.

Later, we may add a feature to detect fake news in Arabic, so we need to normalize and clean the Arabic text.[3]

## 4.4 - Summery extraction

While the (title, author, source, date) of an article can all be gathered from newspaper3k.

To understand what an article is mainly talking about, summaries are a good indicator, which give more meaning.

Summaries are a little tricky to extract but we found that TexRank can do the extractive summary we require. Details on how it works exactly is mentioned in the literature review. After cleaning the body text we feed it to the TextRank algorithm to extract a number of summaries form that text.[2]

For an example, we take this block of text.

'''Santiago is a Shepherd who has a recurring dream which is supposedly prophetic. Inspired on learning this, he undertakes a journey to Egypt to discover the meaning of life and fulfill his destiny. During the course of his travels, he learns of his true purpose and meets many characters, including an “Alchemist”, that teach him valuable lessons about achieving his dreams. Santiago sets his sights on obtaining a certain kind of “treasure” for which he travels to Egypt. The key message is, “when you want something, all the universe conspires in helping you to achieve it.” Towards the final arc, Santiago gets robbed by bandits who end up revealing that the “treasure” he was looking for is buried in the place where his journey began. The end.'''

And the outcome is the following four sentences:

-Santiago is a Shepherd who has a recurring dream which is supposedly prophetic.

-Inspired on learning this, he undertakes a journey to Egypt to discover the meaning of life and fulfill his destiny.

-Santiago sets his sights on obtaining a certain kind of “treasure” for which he travels to Egypt.

-The key message is, “when you want something, all the universe conspires in helping you to achieve it.

We then can apply data cleaning techniques to these four sentences, to obtain the clean summaries. We do this cleaning so the stopword don't affect our ability to compare summaries together with cosine similarity. So the final output would look something like the following:

-Santiago Shepherd recurring dream supposedly prophetic.

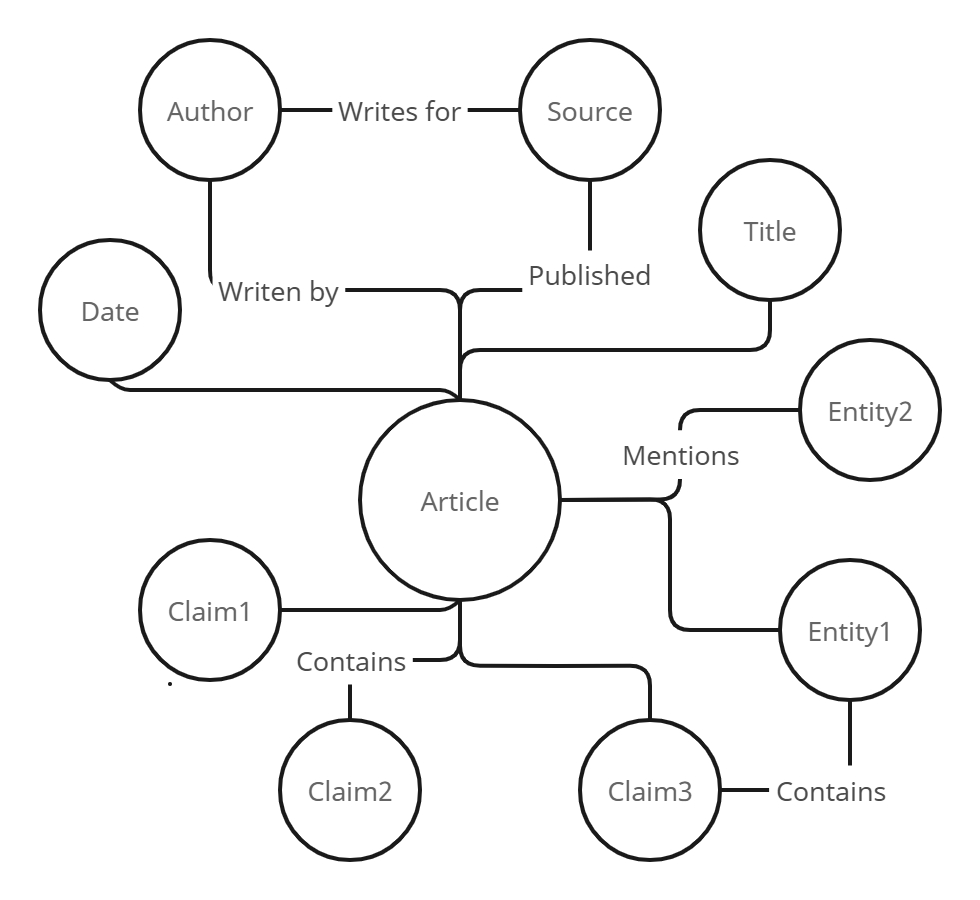
-Inspired learning he undertakes journey to Egypt to discover meaning life fulfill his destiny.

-Santiago sets his sights obtaining certain kind “treasure” for which he travels to Egypt.

-key message “when you want something, all universe conspires helping you to achieve.

## 4.5 – Understanding the structure of our directed nodes

Since we are applying a graphical probability model solution, we’ll need to transform our features into directed nodes and edges, to understand how to input the features into the Bayesian Network.



*Figure - Article graph representation*

*Figure - Node Example Img*

We can first specify the type of nodes we require in order to construct nodes and edges for articles. We may create a node type called "Article" that has the following properties: "title", "source", "author", and "summary", Then, we may create nodes for each article and provide each node the pertinent characteristics.

The relationships between the nodes can then be represented by edges connecting them. As an illustration, we may add an edge to show the relationship "authored by" between an article node and its author node.

Now that we know the structure of our graph, we have two ways to insert and store it into the Bayesian Network, either with Dgraph, or conditional probability tables (CPTs).

While its possible to use Dgraph to store and read from Bayseian Network, they are often represented using specialized data structures and algorithms, which might not be directly compatible with a graph database like Dgraph, however CPT instead of storing the entire join distribution explicitly it represents a large joint probability distribution over many variables in a compact way, which makes it much faster than Dgraph.[2]

## 4.7 – Transforming the string data into Vectors

Probability models like Naïve bayes or Bayesian Network only take class type data in each feature column to predict the target column, which is usually one of two classes. Since our data is string based with each text is somewhat unique, for example no two titles are exactly the same, but they might be similar in context from a human’s point of view.

To solve this challenge, we need to find a way to transform these strings in each column into at least two classes, to do so, we can calculate the cosine similarity of the vector representation of both strings and determine if we can consider them the same class based on a threshold.

Let’s start with the text vectorization. For this we need two types of vectorization models, for known words we use Glove embeddings, and for unknown words we use char Glove. Glove works by comparing the giving word with a list of ready word embeddings each with a 300-dimensionality vector. If the word isn’t in the glove embeddings, we embed each char of that word and combine them to create a 300-dimension vector.

And for our Arabic version we can use the Arabic Glove, and implement the same steps with exception of that the embeddings are of 256-dimensionality.

Now that we have all our data in vector format we can move to classification of these vectors.

## 4.8 – Transforming the Vector data into classes

Our goal is for a user to input a title for example, then our algorithm would find the similar titles in our dataset, then set both the user’s title and the similar title into a class, and anything else into a different class.

To achieve this, we first use cosine similarity to compare each vector in the title column in our dataset with the user’s title, if they are above the threshold of 70%, then set them to ‘A’, if not then set them to ‘B’. So the ‘A’ is meant to indicate that it’s the same class as the user’s class, and the ‘B’ indicates that it is not the same class.

We notice that despite of language this process can enable any language to use the Bayesian Network model.

Now the data is in the right format to be inserted into Bayesian Network model.

## 4.9 - Applying the Graphical Probability model (Bayesian Networks)

We first thought about implementing the Naive Bayes Probability Model for its high performance in studies, but we concluded that naive bayes alone is far too simple to represent our data type. So, we opted for the more complex Bayesian networks approach and we hope to increase the complexity of our graphical model in the future.

Naive Bayes and Bayesian networks are both probabilistic models that use Bayes' theorem to predict class probabilities from feature probabilities. However, there is a difference in the representation of the probabilistic relationships between variables.

**Naive Bayes** is a machine learning algorithm that uses Bayesian principles to model the relationship between a set of features and a binary class label. In this algorithm, each feature is assumed to be conditionally independent of all other features, given the class label. This assumption is known as the "naive" aspect of the algorithm, and it makes the computations relatively simple and efficient.

**Bayesian networks,** On the other hand, are a type of probabilistic graphical model that represent the joint probability distribution of a set of random variables. In a Bayesian network, the variables are represented by nodes in a directed acyclic graph (DAG), and the relationships between the variables are represented by the edges in the graph. The probability distribution is represented by conditional probability tables at each node, which specify the probabilities of the values of the node given the values of its parents in the graph.

So, while Naive Bayes models make the assumption of feature independence, Bayesian Networks models allow for modeling of more complex relationships between variables through the use of a directed acyclic graph.

The basic idea is to model the relationships between these variables as a Bayesian Network.

For example, the network could look like this:

Title -> Summary

Author -> Source -> Summary

Date -> Summary

Here, the arrows represent causal relationships between variables. For instance, the title may influence the summary, and the source may influence both the author and the summary. The date may also have some impact on the summary.

To classify whether a new article is credible or not credible, we would use the network to calculate the probability of the summary given the observed values for the title, author, and source. This would involve using Bayes' theorem to calculate the probabilities of different classes (credible and not credible) based on the probabilities of the features, as well as the relationships between features represented in the network.[2]

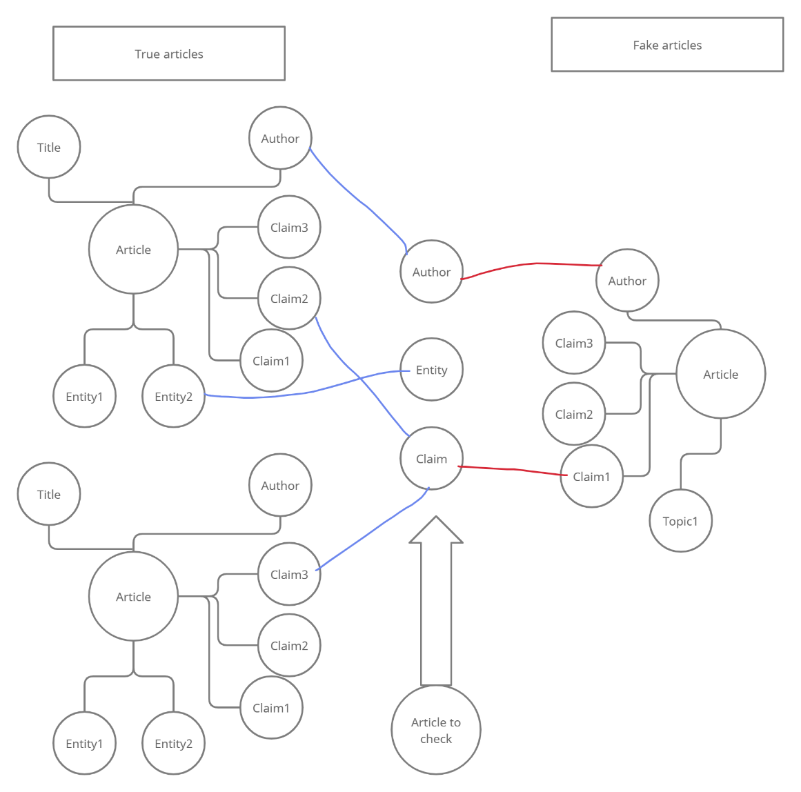
The resulting class probabilities can then be used to make a prediction about the credibility of the article. This prediction could be based on a threshold, for example, classifying the article as credible if the probability of being credible is greater than a certain value, and not credible otherwise.

The resulting probability from the Bayesian networks model can be used as an indicator for the user to determine the credibility of a news article.

To present this information to the user, we can create a report that explains how the model reached its conclusion. For example, the report could include the following information:

1. A brief overview of the Bayesian networks model, including the variables and their relationships as modeled in the network.
2. The inputs used for the new article, including the values for the title, author, source, claim, entity, and date.
3. The resulting probabilities for each class (credible and not credible). This can be presented as a bar graph or table, for example.
4. A description of the decision threshold used to make the prediction, and how it was chosen.
5. A final conclusion about the credibility of the article, based on the probabilities calculated by the model and the decision threshold.
6. A discussion of the strengths and limitations of the model, including the assumptions made and any limitations in the data used to train the model.
7. Feedback and recommendations for further investigation, such as checking additional sources, adjusting existing data, or consulting experts in the fields.

By providing this information to the user, the report can help them understand the reasoning behind the model's conclusion, and provide a basis for further investigation and decision-making.



*Figure 32 - Graph classification visual*

# CHAPTER 5. Future Work

Implementing the current project fully would take a large amount of data to produce reliable results, so we set our first future goal to collect more data and find more efficient ways to do so.

After we assess the reliability of our model, we can improve its complexity by adding more features.

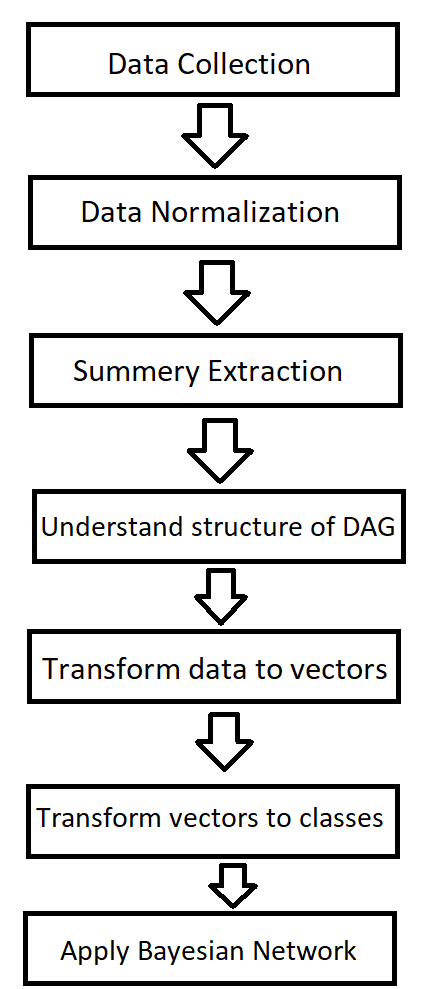
Right now, the threshold for cosine similarity is 70%, but we want to look into how to pick a unique threshold for each feature.

One very important step we should take moving forward is the ability to make use of user feedback, if a large number of the users all agree that the report of a news article is wrong, then some adjustments should be made, this will help the model to learn and iron out any rogue wrong assessments.

# CHAPTER 6. Conclusion

*Figure - System steps*

*Figure - Project Steps*

After our theoretical research and methodology, we found that the steps in this figure would lead to a suitable solution to our problem of providing the user with the probability of a news article being classified as fake or true in our Bayesian Network. And the user's interaction with the system would be as clear and simple as possible.

And we proved that this solution can work with any language that has a word embedding file available, and hopefully this would inspire for more Arabic NLP projects to come out.

# References:

Ahmed, A. A. A., Aljabouh, A., Donepudi, P., & Choi, M. (2021). *Detecting Fake News Using Machine Learning: A Systematic Literature Review*.

Alajramy, L., & Jarrar, R. (2022). Using Artificial Neural Networks to Identify COVID-19 Misinformation. In F. Spezzano, A. Amaral, D. Ceolin, L. Fazio, & E. Serra (Eds.), *Disinformation in Open Online Media* (pp. 16–26). Springer International Publishing. <https://doi.org/10.1007/978-3-031-18253-2_2>

Ben-Gal, I. (2008). Bayesian Networks. In *Encyclopedia of Statistics in Quality and Reliability*. John Wiley & Sons, Ltd. <https://doi.org/10.1002/9780470061572.eqr089>

Brownlee, J. (2020, October 18). Softmax Activation Function with Python. *MachineLearningMastery.Com*. <https://machinelearningmastery.com/softmax-activation-function-with-python/>

*Dgraph Database Overview—Https://dgraph.io/docs/dgraph-overview/*. (n.d.). Retrieved February 13, 2023, from <https://dgraph.io/docs/dgraph-overview/>

Ganesan, K., Zhai, C., & Han, J. (2010). Opinosis: A Graph Based Approach to Abstractive Summarization of Highly Redundant Opinions. In *Coling 2010—23rd International Conference on Computational Linguistics, Proceedings of the Conference* (Vol. 2).

Himdi, H., Weir, G., Assiri, F., & Al-Barhamtoshy, H. (2022). Arabic Fake News Detection Based on Textual Analysis. *Arabian Journal for Science and Engineering*, *47*(8), 10453–10469. <https://doi.org/10.1007/s13369-021-06449-y>

Mihalcea, R., & Tarau, P. (n.d.). *TextRank: Bringing Order into Texts*.

Mughaid, A., Al-Zu’bi, S., AL Arjan, A., AL-Amrat, R., Alajmi, R., Zitar, R. A., & Abualigah, L. (2022). An intelligent cybersecurity system for detecting fake news in social media websites. *Soft Computing*, *26*(12), 5577–5591. <https://doi.org/10.1007/s00500-022-07080-1>

Nastase, V., Mihalcea, R., & Radev, D. R. (2015). A survey of graphs in natural language processing. *Natural Language Engineering*, *21*(5), 665–698. <https://doi.org/10.1017/S1351324915000340>

*Newspaper3k: Article scraping & curation—Newspaper 0.0.2 documentation—Https://newspaper.readthedocs.io/en/latest/*. (n.d.). Retrieved February 13, 2023, from <https://newspaper.readthedocs.io/en/latest/>

*Pydgraph*. (2023). [Python]. Dgraph. <https://github.com/dgraph-io/pydgraph> (Original work published 2016)

Python | Stemming words with NLTK. (2018, October 30). *GeeksforGeeks*. <https://www.geeksforgeeks.org/python-stemming-words-with-nltk/>

*Python String split() Method—Https://www.w3schools.com/python/ref\_string\_split.asp*. (n.d.). Retrieved February 13, 2023, from <https://www.w3schools.com/python/ref_string_split.asp>

*PyTorch documentation—PyTorch 1.13 documentation—Https://pytorch.org/docs/stable/index.html*. (n.d.). Retrieved February 13, 2023, from <https://pytorch.org/docs/stable/index.html>

*Remove non-ASCII characters from a string in Python | bobbyhadz*. (n.d.). Retrieved February 13, 2023, from <https://bobbyhadz.com/blog/python-remove-non-ascii-characters-from-string>

*Removing Stop Words from Strings in Python*. (2020, March 5). Stack Abuse. <https://stackabuse.com/removing-stop-words-from-strings-in-python/>

*tashaphyne.normalize—Https://pythonhosted.org/Tashaphyne/tashaphyne.normalize-module.html*. (n.d.). Retrieved February 13, 2023, from <https://pythonhosted.org/Tashaphyne/tashaphyne.normalize-module.html>

Vatsal. (2023, January 29). *Word2Vec Explained*. Medium. <https://towardsdatascience.com/word2vec-explained-49c52b4ccb71>

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). *Graph Attention Networks* (arXiv:1710.10903). arXiv. <http://arxiv.org/abs/1710.10903>

pgmpy. (n.d.). pgmpy 0.1.19 documentation. Retrieved July 18, 2023, from <https://pgmpy.org/>

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. Stanford University. Retrieved July 18, 2023, from https://nlp.stanford.edu/projects/glove/

# Turn It In Results: