International Journal of Scientific Research & Engineering Trends
Volume 7, Issue 4, July-Aug-2021, ISSN (Online): 2395-566X

Rumor Detection from Social Media

Pragya Gaur, Vasu Verma, Vaibhav Bhardwaj, Utkarsh Kashyap
Department of Computer Science and Engineering,
Meerut Institute of Engineering and Technology,
Meerut 250005, U.P., India.
pragya.gaur@miet.ac.in, vasu.verma.cse.2017@miet.ac.in, vaibhav.bhardwaj.cse.2017@miet.ac.in,
utkarsh.kashyap.cse.2017@miet.ac.in

***Abstract*- We know that now a days the rumours around the world in every filed has become a major concern for everyone that we cannot neglect. Whether you are at home, college, work, or anywhere in the world and you came across something that is known to be a rumour going on around the city, state or country, you start developing that feeling of anxiety and desperation unless you are not clarified on the statement. In order to ensure the right or wrong we go through Internet and search across it to find more and more about it, but sometimes even if we try with multiple search through the google we are not given a clarity about the situation going on around. Hence to overcome this problem we have come up with an Idea of Rumour detection Web App. In this Web App we have designed an app which will give you the result on the basis of an algorithm which will provide you the most releveant content regarding the situation going on around with the exact clarity and verified resources that are available all over the globe.**

***Keywords*- : Rumour Detection, social media, web app.**

I.INTRODUCTION

In the past few years a lot of technologies have emerged as the use of web-scrapping is at the peak in the industry but the issue with these technologies is that they are confined to limited websites and data sets.

So in order to overcome this issue we have proposed a framework that is Designed and integrated in ReactJs while the Backend server has a NodeJs runtime environment. In this we have multiple APIs from the trusted and verified resources integrated to our UX.

Whenever user enters a query that query is sent to the server and then server fetches the data through a RESTful API which returns a response, than we parse that response and returns the desired output.

It will change the whole scenario of rumours going on around at various Social Platforms. Basically we provide a brief overview of research into the rumours going on around that has a goal of classifying rumours on the basis of the data fetched from API.

II. LITERATURE SURVEY

Prior to our Web App there were numerous approaches for this idea. **Jing Ma, Wei Gao and Kam-Fai Wong,** these people from China developed a micro service which does Rumor Detection on Twitter with Tree-structured Recursive Neural Networks. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018.

Another approach was presented by **Cheng Gibson** A Study on Online Rumour Detection on online Social Networks. The main goal of approach was to identify the key traits of rumours on online social networks such as Twitter and Facebook. The importance of automating the identification of rumours is growing ever-increasingly important, given the rise of the internet's popularity as a source of news, and the ever-growing amount of information on the internet.

Then there's this approach by **E. Kochkina, M. Liakata, A. Zubiaga** which is completely written in python This repository contains code for the paper "All-in-one: Multi-task Learning for Rumour Stance classification,Detection and Verification".

Install Prerequisites:

- Python3
- Keras
- Hyperopt
- Optparse

These are some of the approaches which were used prior to this web app and are still used. All of the approaches the same goal of identifying the rumour. But the common thing which all of these researches share is that they classify the rumours and then categorise them accordingly. The need was developed for these approaches when the twitter came into popularity and people start tweeting about false concepts for their own fun. The scope of these researches were pretty good but still the accuracy rate predicted by these researches were not upto the expectation.

It was because there were many challenges such as lack of verified resources. Now a days we have several verified and trusted resources with whom we are able to predict the accuracy of our query. So it's something that is still under process and can be updated very well. It can be done by training a model through Machine Learning as well.

III. BRIEF CONTEXT

The building block of designing this website begins with creating the frontend in ReactJS. We have used class components through the whole project and have define all the necessary data in the state. Moreover we have passed all the data that we have put in state through props in all the components. We have integrated the API on the required components through axios module.

Axios is basically a npm package which comes in react for integrating the APIs. Apart from axios there are multiple dependencies which are used throughout the project. The dependencies that are used at the frontend part are all npm packages as listed below:

- **Styled-components:** Styled-component is a npm package that allows you to define your own tags and it lets you style the Element with the standard css syntax. You need to write your tag in the JSX and you have to define it outside the scope of your functional component.
- **@material-ui styles:** Material UI is a third party platform that provides you with a lot of support into your project not only in styling but also in the components. For ex: Material UI platform consist of lot of built in components like Popover, Dialogue Box, Accordians, ReactPlayer, Tabs and many more.

Briefing about UI, we have mainly three major Components: Header, SideBar and the MainComponent. Header links such as About Us which will redirect you to a page where you can find all the Information about the developers. Beside that you can find a column which says Contact Us there you will find all the details about contacting us and last We have our webapp Title and Logo.

Than we have Sidebar which consist of basically two filters acoording to which you can change the defaults of the rumour set. The first filter provides you the option to filter the Rumour by category, like what kind of rumour are you looking for. While the second filter is all about the trending rumours. Like you can select that if you want to search for a rumour that is going on around and is on trending while the second option lets you choose the time, Like at the specific time there was a rumour going on around.

And finally we have our main component. In the main component there is an input box in which you have to post your query and on the basis of your query you will be provided with the absolute correct answer. We have called an API on the Submit button that is just beside the Input box, On clicking the submit button it will call a regex that will return the data that is relevant to your query and if it doesn't have the expected Answer it will return No data found. All the data that is being interchanged is dynamic and coming and posting through RESTful APIs in nodejs.

The Backend of the Web App is made in Nodejs which is a javascript runtime environment. At the backend we have four APIs. The two APIs are for filtering the Rumour that takes some skip and limit as a input and filters the data on passing the skip and limit the value. Moreover the other two APIs are of POST and GET method. The POST APIs is for storing the query into the MongoDb atlas and the second one is which is of GET method retrieves the data that is relevant to the query. The Regex for filtering the data is defined at the front End only. Every query when posted is assigned a specific Id which is responsible for the retrieval of that specific content.

We have used multiple npm dependencies in this microservice which are as follows:

- **Express:** Express is one of the most popular open source framework for nodejs. It is backed by IBM at present. It provides you flexibility to design your RESTful API structures in a very simplified manner.
- **Body-Parser:** Body-parser is the module that allows the express to read the body and then parse that data into a json object which can be understand by the developers.
- **Mongoose:** Mongoose is the most popular library for connecting your database to your backend. We define our schema under mongoose.
- **Nodemon:** Nodemon is a npm package which is used to automatically update your changes at your server by automatically compiling your code every time you make changes to your code. So you don't need to terminate the server and start it again.

IV. METHODOLOGY

1. Algorithm:

- The algorithm used at the backend does:
- Post Data to the Database
- Get Data from outside source
- Convert and compress the data into loadable json format
- Return the result to given data

2. Algorithm Equation:

```
let searchQuery = [];
if (requestObj.searchKeys && requestObj.searchValue
&& Array.isArray(requestObj.searchKeys) &&
requestObj.searchKeys.length) {
requestObj.searchKeys.map((searchKey) => {
let searchRegex = {"$regex": requestObj.searchValue,
"$options": "-i"};
searchQuery.push({[searchKey]: searchRegex}
```

```
});
requestObj["$or"] = searchQuery; }
```

3. Cloud Hosting Service:

- AWS is used for hosting the micro service APIs
- Create a repository under ECR
- Push your docker image to the ECR
- Create an ECS and put the image into the container
- Under the ECS create a cluster and deploy your Build
- Select ec2 instances for nodes
- Navigate to ec2 and copy the Load Balancer link

4. Result:

- First of all open the load balancer url in browser
- Check the response
- Hit the API through Postman
- Finally integrate into your Project

V. MICRO SERVICE SETUP

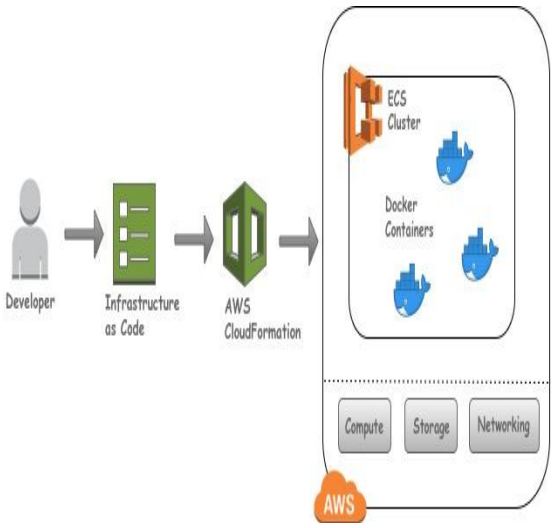

Fig 1. Architecture of Micro Service.

The above diagram shows the architecture for deploying your micro service or web app build to the cloud storage.

VI. EXPERIMENTAL RESULTS

The Client interacts with the UI and post his query to the Input box, then the query is sent to the server as a request. On the basis of parameters passed along with the query it checks for the filtration. After all the above process the response is sent back to the client.

VII. CONCLUSION

The point of this examination was to build a Website for detecting on going Rumours on Social Media. The Rumour detection Website is a real time website which gives you the results on the basis of your query. It verifies the query from the trusted websites and sources and provides you the best result with the maximum accuracy. It is quite easy to operate and user friendly as it is developed in ReactJS.

REFERENCES

[1]. https://docs.aws.amazon.com/cli/index.html
[2]. https://docs.aws.amazon.com/ecs/index.html
[3]. https://expressjs.com/en/5x/api.html
[4]. https://reactjs.org/docs/faq-ajax.html
[5]. https://aws.amazon.com/getting-started/tutorials/deploy-code-vm/
[6]. https://arxiv.org/ftp/arxiv/papers/1911/1911.07199.pdf#:~:text=In%20some%20studies%2C%20rumor%20detection,et%20al.%2C%202018
[7]. https://www.sciencedirect.com/science/article/abs/pii/S0957417418303129
[8]. https://ieeexplore.ieee.org/document/9225567
[9]. https://www.sheffield.ac.uk/dcs/research/our-impact/rumour-detection-social-media
[10]. https://www.hindawi.com/journals/scn/2021/5569064/
[11]. https://link.springer.com/chapter/10.1007/978-3-030-45439-5_38