Cloud-Based System for Fake Tweet Identification

Saranya Krishnan, Min Chen
Division of Computing and Software Systems, School of STEM
University of Washington Bothell
Bothell, USA
{ sarank, minchen2}@uw.edu

***Abstract*—Twitter, as a popular social media platform, has great influence on society. Unfortunately, studies have observed some unreliable or fake information spread via Twitter with harmful consequences. In this paper, we present a cloud-based system to identify tweets with fake news by using tweet text characteristics, machine learning technique, reverse image research, and fake news source comparison. Major components in the system have been deployed as web services, which can be easily integrated or extended in other applications. In addition, web-based interface is developed for general users to validate tweet credibility via web browsers.**

Keywords—fake tweet identification, tweet credibility, cloud computing, web service

I. Introduction

Tweets studied by [3] during the Chile earthquake indicated that rumors propagate in a different way than the actual news. During 2012 Hurricane Sandy, research found that fake images were circulated on Twitter about the disaster [4][5]. There are also arguments and debates on how unreliable news in Twitter and other social media platform may have adverse impact on politics, culture, and business.

In the literature, several techniques have been used to evaluate tweet credibility. For example, [2] uses tweet text characteristics, [1] uses reverse image search, user analysis and crowd sourcing, etc. However, to our best knowledge, there is no generalized end-to-end framework to predict tweet credibility. Different from existing work, a cloud-based system is presented in this paper to validating tweets by checking Twitter user account statistics, text characteristics, media contents, URL links, and using techniques including machine learning, reverse image search, and sentiment analysis. The system allows general users to validate tweet credibility and view associated details via web browsers, and enables researchers and professional developers to integrate and extend the deployed RESTful web services.

II. System Design And Implementation

The system contains two major components: core and website, where core performs application logic and has been deployed as RESTful web services, and website is to interact with general public.

A. Core Architecture

Fig. 1 shows the core architecture. Experiment Runner is responsible for running the whole experiment from receiving user input to generate the model and report based on the model evaluation results. Experiment Orchestrator is responsible for orchestrating the experiment. It receives the tweet ID from the Experiment Runner and send the tweet ID to the Tweet Content Fetcher component. Tweet Content Fetcher is responsible for fetching the tweet content using Twitter API and passes to Feature Set Creator to construct Tweet features.

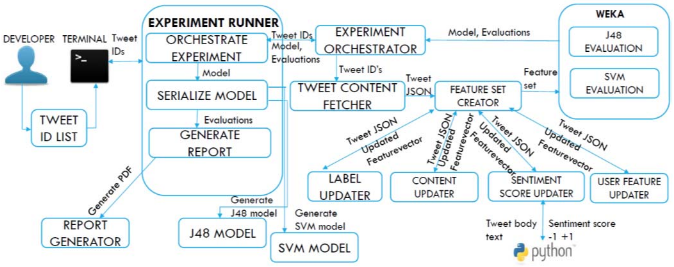

Fig. 1. Core architecture

Feature Set Creator comprises of four updater modules that update corresponding features in the feature set.

1) Label Updater:

To find whether URLs present in a Tweet is credible, Label Updater verifies it against the dataset provided by [4], which contains text and metadata scrapped from 244 websites tagged as "bullshit" by the BS Detector Chrome extension [6]. If there is a match, the Tweet's isUrlCredible feature is set to " No," otherwise to " Yes." Label Updater also verifies whether images present in the Tweet is credible using google reverse image search and google best guess. Google reverse image search uses image as a query and the search engine returns similar ones. Reverse image search can be used to find out where the images comes from. Label Updater retrieves the media entities and sends HTTP requests to Google custom search API with image URL as search query. Google custom search API returns the pages that contains similar images. Label Updater parses the results and checks whether the page source of the returned results is fake by checking the page source against [4] and checks whether the dates are inclined between the page creation date and the tweet creation date. In addition, Google Images has a feature called "Best guess for this image" that returns the textual query matches with the images. Label Updater checks whether the tweet text and Google best guess are related. If any of these searches results in fake, the Label Updater sets the isImageCredible feature as "No," otherwise to"Yes."

2) Content Updater

Content updater is responsible for updating Tweet content features. It receives the Tweet content and retrieves the following features from the Tweet text: number of characters,

978-3-903176-15-7 © 2019 IFIP

number of words, count of question mark, count of exclamation mark, count of uppercase letters, number of hashtags, number of URLs, and number of retweets.

3) Sentiment Score Updater

Sentiment Score Updater sends HTTP request with the tweet text to the sentiment score Python service that calculates and returns the sentiment score. The sentiment score updater then updates the sentimentScore feature accordingly.

4) User Feature Updater

User feature updater receives the tweet content and retrieves the user details for the following features: number of friends, number of followers, friend follower ratio, number of times listed, whether the user has URL, whether the user is a verified user, and total number of tweets posted.

All the feature set are then used in the data mining process. In this study, we used J48 and SVM from WEKA for classification.

B. Website Architecture

Fig. 2 shows the overall website architecture. The website follows Spring model-view-controller(MVC) framework that is designed based on the DispatcherServlet. DispatcherServlet dispatches request to handlers (controllers) along with the configurable handler mappings and view resolution. Default handler is based on the @Controller and @RequestMapping annotations which offers wide range of flexible handling methods. Spring @Controller mechanism is used to create RESTful web sites and applications.

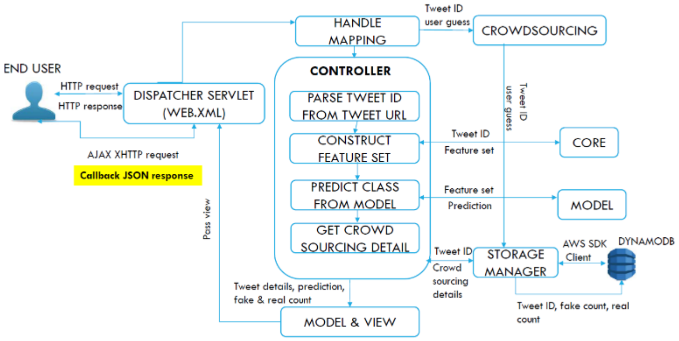

Fig. 2. Website architecture

In brief, Controller receives tweet URL from the user and verifies it via client side validation. If valid, the HTTP request will be sent to DispatcherServlet that in turn dispatches the request to Controller. Otherwise, a proper error message will be displayed to the user. Once the Controller receives the Tweet URL, it parses the tweet URL to retrieve the Tweet ID and send it to Core. Core processes this Tweet ID, constructs a feature set and sends it back to the Controller to be evaluated against the Model as being fake or real. Meanwhile, Controller sends the Tweet ID to the Crowdsourcing component to fetch its fake and real count from Storage Manager. Afterwards, Controller sends the tweet details, predictions and crowd sourcing information to the Model & View component which in turn pass it to DispatcherServlet as HTTP response to be presented.

Fig. 3 shows deployment architecture following continuous integration and continuous deployment (CI/CD) model using AWS services. A distributed version control system (Git) is used in conjunction with bitbucket for version control. Bitbucket pipelines is used for continuous integration and delivery. After every pull request has been merged with the central bitbucket repository, a build using maven is triggered in a docker container maintained by bitbucket pipelines. If all the tests pass and if the build is successful, the build artifacts are zipped and placed in an Amazon S3 bucket. Finally, bitbucket pipelines communicates with AWS CodeDeploy instructing it to deploy a new version of the software whose build artifacts are ready in the S3 bucket.

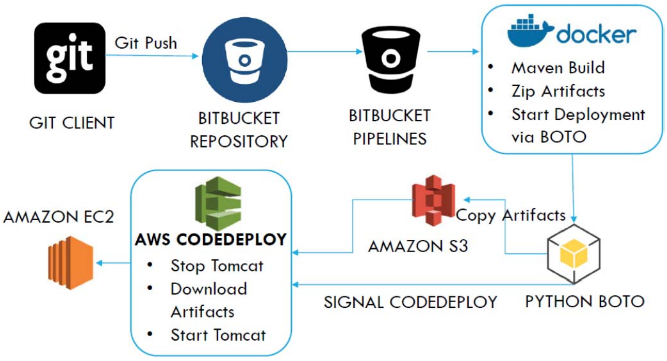

Fig. 3. Deployment Architecture

III. Conclusions and Future Work

This paper presents a cloud-based system to identify tweets that tend to spread fake news and images. In the future, the techniques may be generalized and applied to other social media platform such as Facebook, Instagram, etc. In addition, crowd sourcing data can be fed back to the system to continuously improve the model.

References

[1] A. Gupta, H. Lamba, P. Kumaraguru, and A. Joshi, "Faking sandy: characterizing and identifying fake images on twitter during hurricane Sandy," Proc. 22nd International Conference on World Wide Web, 2013, pp. 729-736.

[2] H. Hamdan, P. Bellot, and F. Bechet, "Lsislif: Feature extraction and label weighting for sentiment analysis in twitter," Proc. 9th International Workshop on Semantic Evaluation, Association for Computational Linguistics, 2015, pp. 568-573.

[3] A. L. Hughes, and L. Palen, "Twitter adoption and use in mass convergence and emergency events," Inter. J. Emergency Management, vol. 6, no. 3-4, pp. 248-260, 2009.

[4] M. Risdal, Getting Real about Fake News. https://www.kaggle.com/mrisdal/fake-news.

[5] D. Saez-Trumper, "Fake tweet buster: a webtool to identify users promoting fake news on twitter," Proc. 25th ACM conference on Hypertext and Social Media, 2014, pp. 316-317.

[6] D. Sieradski, BS Detector. https://github.com/selfagency/bsdetector.