DASS Report

Scrapshut
Team no: 24
Members:

Avani Gupta(2019121004)
Parv Joshi (2018101062)
Siddharth Jain(2018101038)
Apoorva Tirupati(2019121012)

Brief Problem Statement:

To develop a web platform for ScrapShut where users can come and rate different URLs and get reviews of other users on a particular URL. This web app aims to provide users with data on the credibility of the content provided by various websites, to curb fake news perpetration.

Project Overview:

We predict the genuineness of a website based on the majority of user ratings, and ML model trained using features like website's scraped data, user's ratings/ reviews data from the website. Users further would be asked to tell which portions of the article made them think the article is fake and provide basis for their report . We aim to provide users with data on which websites are genuine and not genuine and protect the users malicious websites and fraud.

Development Environment

■ Editor - VS Code
■ Web framework - Django 3
■ Collaboration tools - Gitlab
■ Frontend - Angular.js
■ Documentation - Google Docs
■ Database – AWS, Heroku
■ Machine Learning and Web Scraping
○ Language: Python
○ Framework: Jupyter Notebook

Users profile

- Common users using the internet who want to get reviews on a site/article hosted online.
- Users doubtful about the genuineness of a website and who want to check whether they are'nt entering a fraud website.

Project Plan

Milestone Schedule

Milestone	Due Date	Release	Deliverable?
Requirements Study ● Meeting with client ● Requirements gathering	21 Jan	Sprint1	No

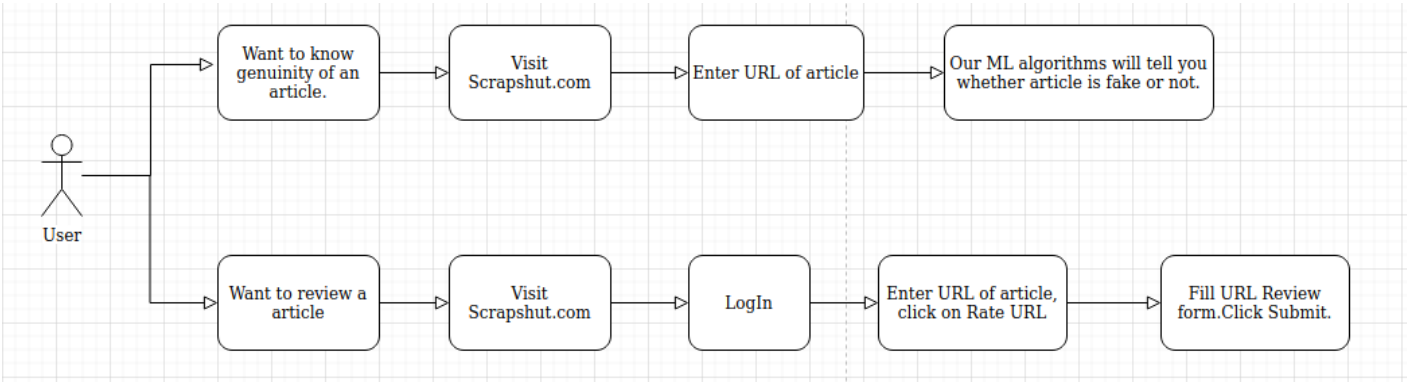
● Feasibility Study ● Project Scope Documentation ● Project Concept Documentation			
Finalizing requirements ● Project Plan ● SRS creation ● Study of ML models for Fake Content Detection ● Study of Web Scraping Techniques	1 Feb	Sprint 2	Yes
● Sprint Goal Document ● Learning Vue.js, Django ● Basic UI interface (Vue.js)	15 Feb	Sprint 3	Yes
Basic Website Creation ● Frontend(Angular.js) and backend(Django) ○ User login/signup ○ User Authentication ○ Upload link of article ○ URL verification of article link ○ Option to Rate and Review the article	25 - 29 Feb	Sprint 4 R1	Yes
Website Enhancement ● Option to mark a website as fake ● Enhancement of UI	12 - 14 Mar	Sprint 5	Yes

● Improve upon security aspects of Authorisation *Basic ML model* ● Gathering Dataset ● Feature Engineering ● Training basic ML model for fake news detection Scraping ● Scraping article title, body from link posted by users			
Building different ML models ● Training different ML models ● Building their performance metrics ● Trying Deep Learning models, Neural Networks Website backend Improvement ● Enhance User dashboard	23 March	Sprint 6	Yes
● Updated Design Document ● Integration of ML model with Website ● Deployment ○ ML model in AWS ○ Database in AWS and Heroku	11 Apr	Sprint 8 R2	Yes

Team Communication

- Meet on a weekly basis.
- Contact each other via messenger, Whatsapp, email.
- Contact with Client via Email and Whatsapp group

General Flow Diagram(How Scrapshut works)



Features Implemented:

- Complete WebApp Is Responsive
- Signup
    - Email must be valid
    - Strong password(One uppercase ...)
    - OTP Email verification
    - Unique Username is required
    - Unique Email is required
- Homepage
    - Added Info About Scrapshut Org
    - Added A Contact Form So User can send query to Admin
    - Added Small Picture Buttons for Insta, Pinterest, FB and later on href can be added to access their page.
- StartPage (Logged In Page)
    - Logout Hyperlink
    - Dashboard Button
    - Rate And Check Website Button
    - Contact Form
    - Small Social Media Buttons that can be hyperlinked
- Login
    - Valid set of credentials required for logging in.
    - Once Left Logged In Page Cannot Come Back
- Check URL Genuineness
    - URL verification
        - URL entered must exist
    - Scraping
        - Title, body and all associated links scraped and stored.
    - Prediction
        - Prediction based on User reviews for that URL and ML model

- Rate URL
    - URL Verification
        - URL entered must exist
    - Review form

    - If more than some threshold number of users mark a website as fake it goes in a fake URLs database which is used to display the genuineness
- Dashboard Displays All Past Rated History OF Logged In User

Challenges Faced

Frontend/Backend Development:

- We dealt with a lot of changes in the frameworks, libraries being used as the client was shifting base and had to scrap much of our initial work
- Setting up the environment to be ready to code and build consumed a lot of our time
- We are dabbling in web development for the very first time and had a steep learning curve with respect to learning about all the various tools and frameworks out there
- We had reached the CORS Header error because our angular could not connect to our backend. We had to whitelist is in Django Framework File (Setting.py)
- CORS Error also occurred during URL Verification as frontend had not access to do get requests from other websites all had to be done in Back-End.
- One Challenge was to link the DJANGO Backend to an email account which can be used to receive and send emails.
- To make a more user friendly interface, we added popups for review form, dashboard and rating so that users don't need to go to a new page.
- Adding various security features was another task like OTP Authentication, Making Sure Once Left Page Always Logged Out, and maintaining their error handling.
- Selection Of Colours And Design Was Also another challenge.
- Making The WebApp Responsive was also a challenge at some points.

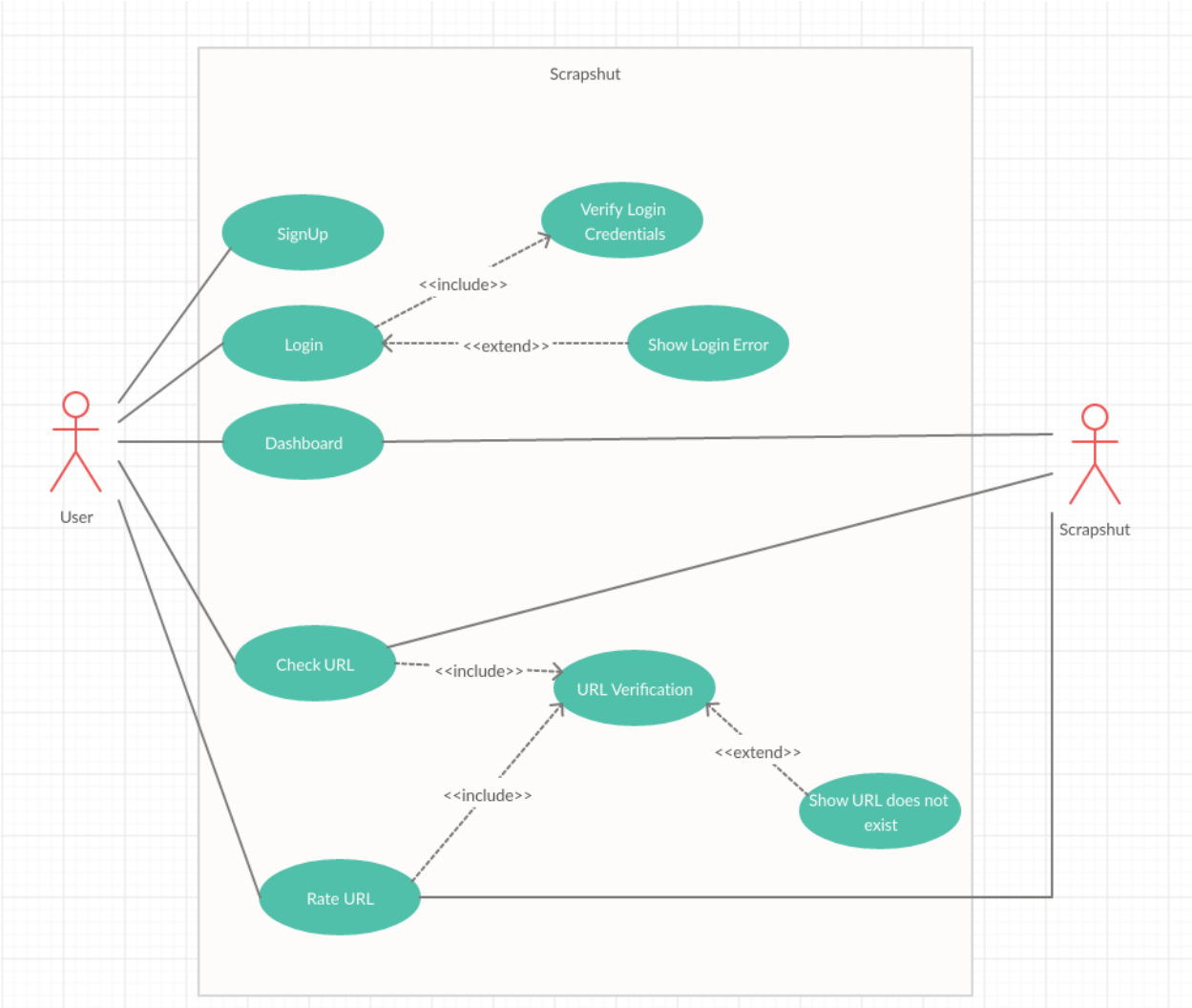
Training Models:

- The models took a lot of training time and memory and jupyter-notebook kernel used to die due to it.
- The biggest challenge faced was prediction in real time based on static datasets. After a lot of survey came up with the idea of Online learning and trained based on that, Still we need more real time user reviews in order to predict with greater accuracy.
- Hence trained lstm on Google Collab which also took a lot of time, but lesser than local system.
- Later made some optimizations which reduced training time.
- The memory remained a problem, hence trained the models in batches.

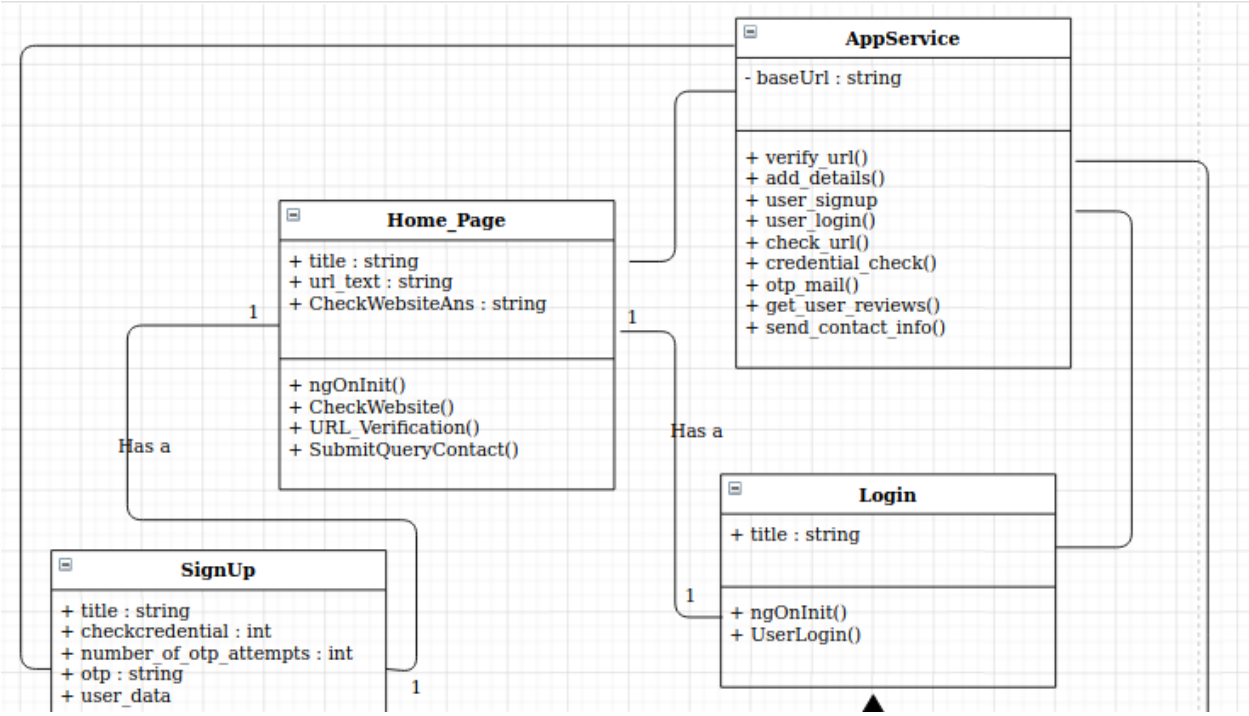
- Now came, dataset, we were not having a dataset of user reviews initially so searched and gathered a dataset containing article links along with corresponding labels- "Real" / "Fake" and trained the models on them.
- HyperTuning the model parameters to make it real time was a good challenge faced.
- Deciding upon different datasets to make the model generalise better was another.
- Still, we don't have much user reviews data which is a hindrance to prediction in real time. As more data is gathered the models would improve.
- RAM Of Laptops Was Not Good Enough For Training Models With Large Amounts Of Data. This created many problems.

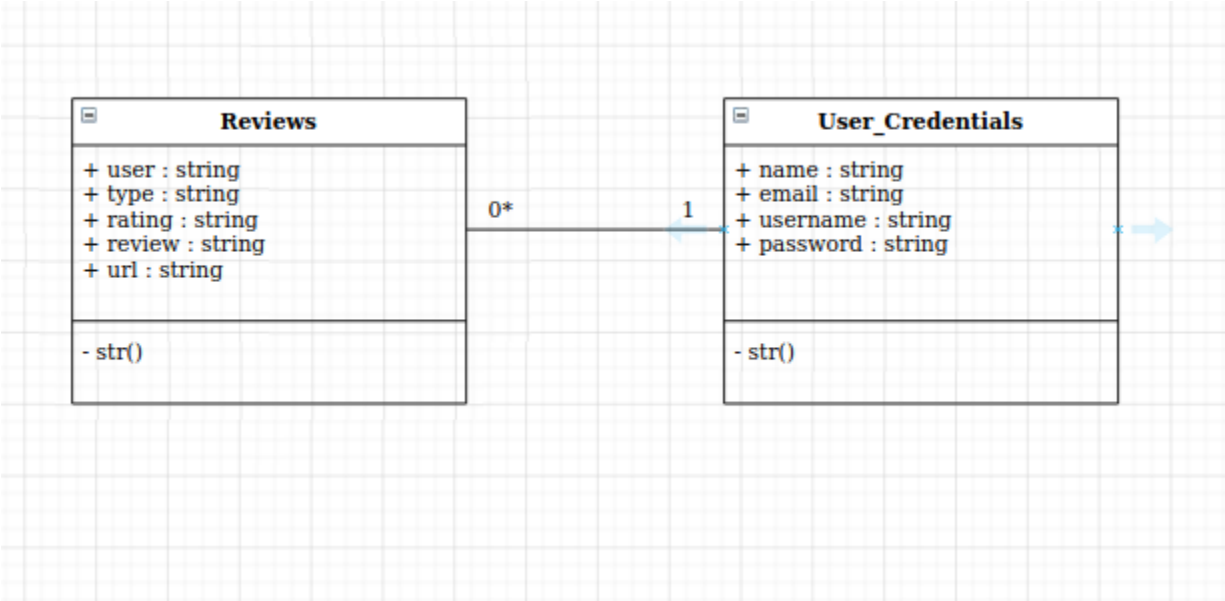
Diagrams

UML Use Case Diagram



UML class diagram





UML SEQUENCE DIAGRAMS:

For Writing Review:

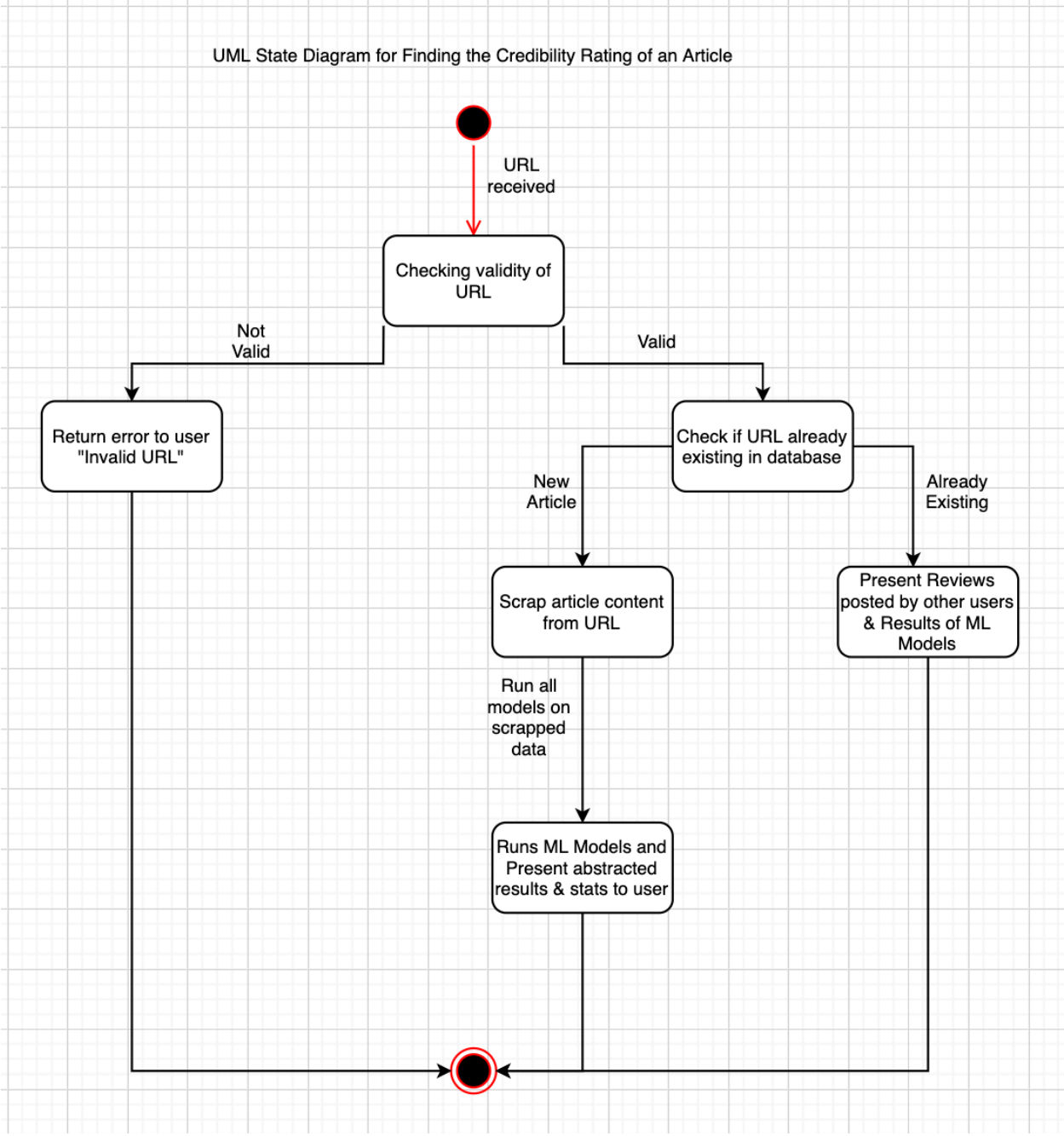
SequenceDiagram1

For Checking Review:

SequenceDiagram2

UML State Diagram

(For Finding Credibility Rating of an URL)



Project System interfaces

User Interface

Users can use this system in two ways.

1) Checking authenticity of an article
2) Writing Reports/Giving Reviews/Ratings

Model of Project

User

Responsible for all the functions and data related to the user object, which includes their login credentials, their reviews/reports against articles/news sources & their saved articles

- The methods an object of the user class can implement are very simple:
- 1. Can report an Article via the Report class's object
- 2. Can view the reports related to an article
- 3. Can run the DL model on an unseen article they're doubtful about

Article

Class state

- What information is the class responsible for maintaining?
- Content of Article
- Newspaper that published it
- Article writer
- Lists of reports and reviews on it

Class behavior

- Add to DL training set

NewsSource

Class state

- Credibility Rating wrt News Section
- Reviews written on the News Channel

Report

Class state

- Explanations as to why they found it fake
- Sources/Citations as proof to base their opinions if possible

Design Rationale

1. We initially had to take a call on whether our Deep Learning Model that detects fake articles will be one that trains on-the-fly or in-batches. After discussion with the client and considering the cost of one-the-fly versus batch-trained, we arrived at choosing batch-trained as it would be less expensive over of having a constant training occurring at all times, instead on attaining an additional 100 new samples, we'd retrain the model.

2. Another major reason for choosing batch-training is for consistency in results. With on-the-fly training happening in the background at all times (like a perpetual learning machine), the results are subject to variability and volatility at the smallest intervals, this cannot be explained for, and makes the user question the site's or react.js, ones we had experience with from our assign credibility.

3. We initially used a basic word-to-vec single-layered neural network, on a simple fake-not fake annotated news dataset, to learn how to implement deep learning and get our basics clear and then shifted to a more complicated CNN, using Glove, and trained on a better more credible Kaggle Dataset

3. We've learnt a lot in the process about ML & Neural Networks and we finally built four models, LSTM, CNN, xgboost & a passive aggressive

classifier. We learnt a lot from the various results, and decided to make an ensemble model based on the 4 results. But after multiple trials, decided not to, as it would give a skewed understanding and felt four individual results would have more use, as they explain in 4 different aspects.

ML Models

Overview of Features & Development Process :

Features:

Scraping

Tech stack: beautiful soup library
request library

The scraper scrapes article title, body, all links in the article from url using beautifulsoup and request library.

URL based prediction
A database is dynamically created containing all the reviews of users for different urls.
It includes url, UserReview, flag for fake news
If there are more than 20 reports for a particular website the website url gets added to this database.

ML model

Model1: Various models on Kaggle fake news dataset

Dataset: Kaggle fake news

There are various models trained on kaggle fake news net.
The dataset contains two files train.csv and test.csv.
train.csv: A full training dataset with the following attributes:

- id: unique id for a news article

- title: the title of a news article
- author: author of the news article
- text: the text of the article; could be incomplete
- label: a label that marks the article as potentially unreliable
    - 1: unreliable
    - 0: reliable

test.csv: A testing training dataset with all the same attributes at train.csv without the label.

One entry of dataset looks like:

train.csv (94.06 MB)

	id	title	author	text	label
1	0	House Dem Aide: We Didn't Even See Comey's Letter Until Jason Chaffetz Tweeted It	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Letter Until Jason Chaffetz Tweeted It By Darrell Lucus on October 30, 2016 Subscribe Jason Chaffetz on the stump in American Fork, Utah (image courtesy Mic...	1
2	1	FLYNN: Hillary Clinton, Big Woman on Campus - Breitbart	Daniel J. Flynn	Ever get the feeling your life circles the roundabout rather than heads in a straight line	0

Data Preprocessing and embeddings creation

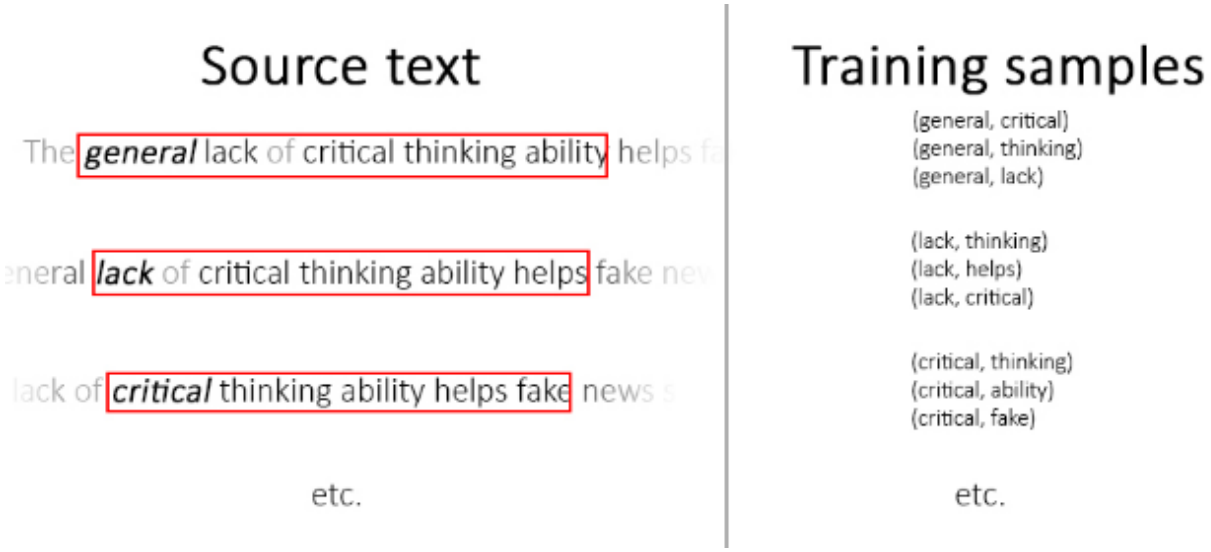
All null values entries in the dataset were removed, stopwords were removed and data was tokenized, along with some other minor preprocessings.

Building word embedding matrix.

Data used: glove

Building a word embedding matrix is done using a *shallow neural network* with a bottleneck in the middle. Data generation for this task is done by mining a large body of text for word-target pairs. For each word in the text, a series of word pairs are generated by sampling a random target word within a window.

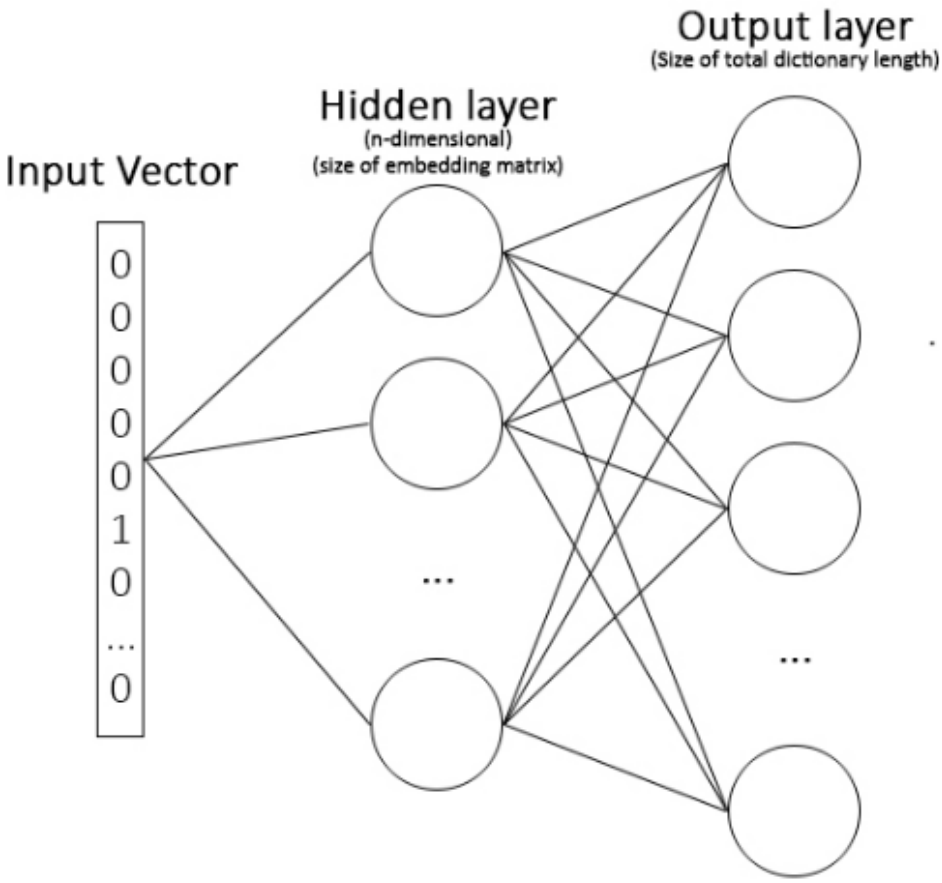
Consider the sentence "*The general lack of critical thinking ability helps fake news spread like cancer*". Generating word-pairs is usually done with a window function, meaning for each word in the body of text one or several target words are selected from the surrounding words (let's say 5). Generating word pairs could then look something like this:



To train the network we create a keyed dictionary. The input is a one-hot vector the same size as the dictionary, with all zeros except for the index location of the input word in the dictionary. Using softmax means the network outputs probabilities for each word given as an input. The network also has a single hidden layer of the

neurons equal to the number of words in the embedding layer. It is trained on the word pairs.

Once training is complete the output layer is removed and the embedding matrix is made from weights of the hidden layer.



Architecture of network

After training we get a network that encodes for the individual word vectors. Each time we present the neural net with an input vector encoding a word, the corresponding weights in the hidden layer are selected.
Each word in the dictionary has an associated n-dimensional vector attached. To get a word vector, we only need to find its index in the dictionary and cosine similarity is used to find related words.

Incorporating such an embedding matrix in a deep learning architecture for other tasks is a form of transfer learning. In practice

this translates to much lower time and data requirements to fit the network properly.

Models trained

Trained many models like CNN, LSTM, xg-boost, logistic regression, decision trees(with criterion entropy index, gini index, extra trees classifier), Random forces, KNN

Out of all models CNN, LSTM, XGBoost are performing the best as expected.

Now the various hypertuning of parameters and metrics achieved are as follows:

CNN

Experimented with different epochs out of which 25 provides the best results.
Experimented different layers and regularization parameters like dropout etc to prevent overfitting.
Out of which the best result was obtained in

```
def cnn_model(sequence_input, embedded_sequences, classes=2):
    x = Conv1D(64, 5, activation='relu')(embedded_sequences)
    x = MaxPooling1D(5)(x)
    x = Conv1D(128, 3, activation='relu')(x)
    x = MaxPooling1D(5)(x)
    x = Conv1D(256, 2, activation='relu')(x)
    x = GlobalAveragePooling1D()(x)
    x = Dense(2048, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
    preds = Dense(classes, activation='softmax')(x)

    model = Model(sequence_input, preds)
    return model
```

LSTM

Experimented with different epochs on different architectures out of which 5 provides the best results.

Experimented different layers and regularization parameters like dropout etc to prevent overfitting/

The lstm giving best metrics was:

```
MAX_SEQUENCE_LENGTH=1500
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedded_sequences = embedding_layer(sequence_input)
lstm_model = LSTM_model(sequence_input, embedded_sequences, classes=2)

lstm_model.compile(loss='categorical_crossentropy',
              optimizer='adamax',
              metrics=['acc'])

print(lstm_model.summary())

lstm_model.fit(train_data, train_labels,
          validation_data=(valid_data, valid_labels),
          epochs=5, batch_size=64)
```

```
Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 1500)              0
_________________________________________________________________
embedding_1 (Embedding)      (None, 1500, 300)         49661400
_________________________________________________________________
lstm_7 (LSTM)                (None, 1500, 32)          42624
_________________________________________________________________
lstm_8 (LSTM)                (None, 1500, 64)          24832
_________________________________________________________________
lstm_9 (LSTM)                (None, 128)               98816
_________________________________________________________________
dense_7 (Dense)              (None, 4096)              528384
_________________________________________________________________
dropout_3 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_8 (Dense)              (None, 1024)              4195328
_________________________________________________________________
dense_9 (Dense)              (None, 2)                 2050
=================================================================
Total params: 54,553,434
Trainable params: 4,892,034
Non-trainable params: 49,661,400
_________________________________________________________________
None
Train on 9000 samples, validate on 500 samples
Epoch 1/5
9000/9000 [==============================] - 333s 37ms/step - loss: 0.4674 - acc: 0.7714 - val_loss: 0.4031 - val_a
cc: 0.8060
Epoch 2/5
9000/9000 [==============================] - 356s 40ms/step - loss: 0.3157 - acc: 0.8630 - val_loss: 0.2449 - val_a
cc: 0.9040
Epoch 3/5
9000/9000 [==============================] - 318s 35ms/step - loss: 0.2541 - acc: 0.8980 - val_loss: 0.3500 - val_a
cc: 0.8660
Epoch 4/5
9000/9000 [==============================] - 379s 42ms/step - loss: 0.2164 - acc: 0.9139 - val_loss: 0.2769 - val_a
cc: 0.8900
Epoch 5/5
9000/9000 [==============================] - 388s 43ms/step - loss: 0.1965 - acc: 0.9230 - val_loss: 0.2075 - val_a
cc: 0.9060

<keras.callbacks.callbacks.History at 0x7f7a9eb71f90>
```

Other architecture which performed similar was:

```
# LSTM Neural Network
lstm_model = Sequential(name = 'lstm_nn_model')
lstm_model.add(layer = Embedding(input_dim = max_features, output_dim = 120, name = '1st_layer'))
lstm_model.add(layer = LSTM(units = 120, dropout = 0.2, recurrent_dropout = 0.2, name = '2nd_layer'))
lstm_model.add(layer = Dropout(rate = 0.5, name = '3rd_layer'))
lstm_model.add(layer = Dense(units = 120,  activation = 'relu', name = '4th_layer'))
lstm_model.add(layer = Dropout(rate = 0.5, name = '5th_layer'))
lstm_model.add(layer = Dense(units = len(set(y)),  activation = 'sigmoid', name = 'output_layer'))
# compiling the model
lstm_model.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
```

Table for metrics

Model	Accuracy	Precision	Recall	F1 score
CNN	0.958000	0.973451	0.936170	0.954447
LSTM	0.934000	0.926070	0.944444	0.935167
XGBoost	0.952000	0.956710	0.940426	0.948498
Decision Tree with Gini Index	0.734000	0.710744	0.731915	0.721174
Decision Tree with criterion entropy index [DT(E)]	0.716000	0.697872	0.697872	0.697872
KNN	0.552000	0.536424	0.344681	0.419689
Random Forest	0.716000	0.666667	0.791489	0.723735
Linear Regression	0.622000	0.602679	0.574468	0.588235

Entra Tress Classifier	0.617241	0.581699	0.654412	0.615917

New Feature: Find related URLs
Tech used: Beautiful soup
googlesearch search module

User can give the topic name he wants to read on and he will get the urls of corresponding topics from trusted sources.

Keyword extraction
Extract keywords from article body using tf-idf
article body taken, stopwords removed, tf-idf calculated, score calculated for words,top k most scored words returned.

Model 2: Passive Aggressive classifier and online learning for incorporating user reviews

Tech used: sklearn

Model trained using tf-idf and passive aggressive classifier which is an online learning algorithm.

Passive aggressive classifier as the name suggests remains passive for a correct classification outcome, and turns aggressive in the event of a miscalculation, updating and adjusting. Unlike most other algorithms, it does not converge. Its purpose is to make updates that correct the loss, causing very little change in the norm of the weight vector.
dataset: Kaggle Getting real about fake news
Size: 7796×4
One dataset looks like:

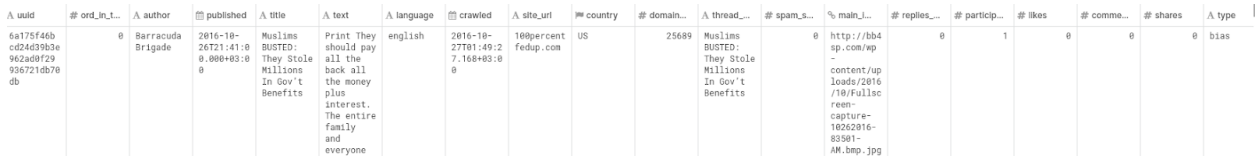
	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE

Accuracy achieved in classifying real/fake news : 92.58%

Further training and Choosing the best model
LSTM

Further enhanced LSTM and trained the one with best architecture from various experimentations.
Trained LSTM on various other datasets:

[Kaggle getting Real about fake news]



uuid	ord_in_t...	author	published	title	text	language	crawled	site_url	country	domain...	thread_...	spam_s...	main_i...	replies_...	particip...	likes	comme...	shares	type
6a175f46bcd24d39b3e962ad0f29936721db70db	0	Barracuda Brigade	2016-10-26T21:41:00.000+03:00	Muslims BUSTED: They Stole Millions In Gov't Benefits	Print They should pay all the back all the money plus interest. The entire family and everyone	english	2016-10-27T01:49:27.168+03:00	100percentfedup.com	US	25689	Muslims BUSTED: They Stole Millions In Gov't Benefits	0	http://bb4sp.com/wp-content/uploads/2016/10/Fullscreen-capture-10262016-83501-AM.bmp.jpg	0	1	0	0	0	bias

[Kaggle Fake News detection]

	URLs	Headline	Body	Label
1	http://www.bbc.com/news/world-us-canada-41419190	Four ways Bob Corker skewered Donald Trump	Image copyright Getty Images On Sunday morning, Donald Trump went off on a Twitter tirade	1

These datasets were chosen keeping in mind the diversity to prevent model from restricting to a particular dataset and generalise more.
Various number of epochs were tried and the model performed best on 5 epochs.

The metrics on test dataset were:
Accuracy: 0.940000
Precision: 0.978723
Recall: 0.901961
F1 score: 0.938776

Models predictions on real time scraped urls increased drastically and we knew its a right track. Furthur the models's performance improves by user reviews and BOOM! it gets the ability to predict accurately on any article source.

Testing

Process of testing:

- Setting up the test for a particular piece of an application (called the system under test)
- Performing the actual testing (interacting with the system under test)
- Observing the resulting behavior and checking whether expectations were met.

Testing Approach and Types of testing:

Functionality Testing

- Test the outgoing links from all the pages to the specific domain under test.
- Test all internal links.
- Test links jumping on the same pages.
- Test links used to send email to admin or other users from web pages.
- Finally, link checking includes, check for broken links in all the above-mentioned links.
- First, check all the validations on each field.
- Check for default values of the fields.
- Wrong inputs in the forms to the fields in the forms.
- Options to rate website and check website and url verification.
- LocalMemory and HTML/CSS testing.
- Testing if the Email ID for the user verification using the local host itself are correct or not.
- Are We Able To Scrape Website
- Are We Able to get Output form ML Model

Usability Testing

- The website should be easy to use.
- The instructions provided should be very clear.
- Check if the instructions provided are perfect to satisfy its purpose.
- The instructions should be readable enough for the user.
- The Navigation Bar should be provided on homepage and start page.

- The main menu basically should contain the following:
    - Logout button on each page for usability testing of the user.
    - Dashboard
- It should be consistent enough.
- Content checking.
- Navigation flow checking.
- Check If Popups Are Coming Easily for Dashboard, Rate URL and Check URL

Interface Testing

- Check if all the interactions between these servers are executed and errors are handled properly.
    - Web server and application server interface
    - Application server and Database server interface.
    - Application server and Gmail server interface.
    - Terminal (ng serve) and local host interface.
- If the database or web server returns an error message for any query by the application server then the application server should catch and display these error messages appropriately to the users.

Integration testing

- Integration of login with dashboard. Whether the correct dashboard is displayed according to account type.
- Integration of the Logout with Sign in.
- Integration of ML Modals with buttons and scraper was tested.
- Integration of Dashboard Table with user data on start page.
- Integration of Check Website on Home Page.
- Integration Of Rate And Check Website On Home And Start Page.
- Integration Of OTP and Mail Feature with contact and signup form

Testing Tools and Environment

- Automation and Test management tools needed for test execution.
- Figure out the number of open-source as well as commercial tools required, and determine how many users are supported on it and plan accordingly testing Tool.

Testing Tools:

- Postmanis a powerful tool for performing integration testing with your API. It allows for repeatable, reliable tests that can be automated and usedin a variety of environments and includes useful tools for persisting data and simulating how a user might actually be interacting with the system.Postman was used to test and check the validity of the various get and post requests.
- Jasmine (Testing Server For Angular)
- Selenium (For Django)
- Manual Checking

Testing Environment:

For the test environment, a key area to set up includes:

- System and applications - Gitlab , Text editors - VSCode ,npm
- Database server - SQLite
- Front-end running environment – AngularJS
- Back-end running environment - DJANGO
- Client operating system - Windows/Linux
- Browser- Google Chrome/Mozilla Firefox
- Hardware includes Server Operating system
- Network - Localhost (frontend running on port 4200, backend running on port 8000)

Testing Summary:

- We performed load testing to check site performance under normal & peak conditions.
- We did unit testing of all frontend & backend components,including our deployed ML models.
- We did security testing for user authentication & verification.
- We did localization testing by trying out the site on various browsers & platforms.
- We did interface testing of the entire UI and have tried to keep things as easy to understand as possible.
- We did integration testing after connecting the 3 different components of our project - the UI, database & the on-the- fly