**Source code**

Main.py

from flask import Flask, render\_template, request, jsonify

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import RegexpTokenizer, sent\_tokenize

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import load\_model

import numpy as np

nltk.download('stopwords')

app = Flask(\_\_name\_\_)

model = load\_model('fake\_review.h5') # Load the model at the start to save loading time per request.

def preprocess(reviews):

stop\_words = set(stopwords.words("english"))

tokenizer = RegexpTokenizer(r'\w+')

processed\_reviews = []

for review in reviews:

words = []

sentences = sent\_tokenize(review.lower())

for sentence in sentences:

tokens = tokenizer.tokenize(sentence)

filtered\_words = [w for w in tokens if w not in stop\_words and len(w) > 1]

words.extend(filtered\_words)

processed\_reviews.append(words)

return processed\_reviews

def convert\_text\_to\_no(reviews):

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(reviews)

sequences = tokenizer.texts\_to\_sequences(reviews)

maxlen = 100

return pad\_sequences(sequences, maxlen=maxlen)

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

review = request.form['review']

processed\_review = preprocess([review])

sequence\_review = convert\_text\_to\_no(processed\_review)

sequence\_review = np.array(sequence\_review, dtype=np.float32)

prediction = model.predict(sequence\_review)

result = "Original Review" if prediction[0] >= 0.2 else "Fake Review"

return jsonify(result=result)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

script.js

from flask import Flask, render\_template, request, jsonify

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import RegexpTokenizer, sent\_tokenize

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import load\_model

import numpy as np

nltk.download('stopwords')

app = Flask(\_\_name\_\_)

model = load\_model('fake\_review.h5') # Load the model at the start to save loading time per request.

def preprocess(reviews):

stop\_words = set(stopwords.words("english"))

tokenizer = RegexpTokenizer(r'\w+')

processed\_reviews = []

for review in reviews:

words = []

sentences = sent\_tokenize(review.lower())

for sentence in sentences:

tokens = tokenizer.tokenize(sentence)

filtered\_words = [w for w in tokens if w not in stop\_words and len(w) > 1]

words.extend(filtered\_words)

processed\_reviews.append(words)

return processed\_reviews

def convert\_text\_to\_no(reviews):

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(reviews)

sequences = tokenizer.texts\_to\_sequences(reviews)

maxlen = 100

return pad\_sequences(sequences, maxlen=maxlen)

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

review = request.form['review']

processed\_review = preprocess([review])

sequence\_review = convert\_text\_to\_no(processed\_review)

sequence\_review = np.array(sequence\_review, dtype=np.float32)

prediction = model.predict(sequence\_review)

result = "Original Review" if prediction[0] >= 0.2 else "Fake Review"

return jsonify(result=result)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Fake Review Detector</title>

<!-- Bootstrap CSS -->

<link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

<!-- Animate.css -->

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.0.0/animate.min.css"/>

<style>

body, html {

height: 100%;

margin: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: #f5f7fa;

color: #333;

}

.bg-gradient {

background-image: linear-gradient(to right top, #65dfc9, #6cdbeb);

height: 100%;

width: 100%;

position: fixed;

top: 0;

left: 0;

z-index: -1;

}

.container {

padding-top: 10%;

}

.btn-primary {

background-color: #6cdbeb;

border: none;

}

.btn-primary:hover {

background-color: #65dfc9;

border: none;

}

.form-control {

border-radius: 0.25rem;

border: 1px solid #ced4da;

}

.form-control:focus {

box-shadow: 0 0 0 0.2rem rgba(108, 221, 235, 0.25);

border-color: #6cdbeb;

}

#reviewForm {

background: white;

padding: 30px;

border-radius: 8px;

box-shadow: 0 6px 20px rgba(0,0,0,0.15);

}

#loader {

display: none; /\* Hidden by default, shown during AJAX request \*/

text-align: center;

color: #6cdbeb;

}

</style>

</head>

<body>

<div class="bg-gradient"></div>

<div class="container">

<div class="row justify-content-center">

<div class="col-md-8">

<h1 class="text-center mb-4">Fake Review Detector LSTM</h1>

<form id="reviewForm" class="needs-validation" novalidate>

<div class="form-group">

<textarea id="reviewText" class="form-control" rows="5" placeholder="Enter your review here..." required></textarea>

<div class="invalid-feedback">Please enter a review.</div>

</div>

<div class="text-center">

<button type="submit" class="btn btn-primary">Check Review</button>

</div>

<div id="loader">Fetching result...</div>

<div id="result" aria-live="polite" class="mt-3"></div>

</form>

</div>

</div>

</div>

<script>

document.addEventListener('DOMContentLoaded', function() {

const form = document.getElementById('reviewForm');

const reviewText = document.getElementById('reviewText');

const loader = document.getElementById('loader');

const result = document.getElementById('result');

form.addEventListener('submit', function(e) {

e.preventDefault();

loader.style.display = 'block'; // Show loader

fetch('/predict', {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

},

body: `review=${encodeURIComponent(reviewText.value)}`

})

.then(response => response.json())

.then(data => {

result.innerHTML = `<span class="firework">🎆</span> ${data.result} <span class="firework">🎆</span>`;

result.className = 'alert alert-success mt-3 animate\_\_animated animate\_\_fadeIn';

loader.style.display = 'none'; // Hide loader

})

.catch(error => {

console.error('Error:', error);

loader.style.display = 'none'; // Hide loader

result.textContent = 'Error fetching results';

result.className = 'alert alert-danger mt-3 animate\_\_animated animate\_\_fadeIn';

});

});

});

</script>

</body>

</html>

Model training code:

import pandas as pd

df=pd.read\_csv(r'/kaggle/input/fake-review/fake reviews dataset.csv')

'''

replace the column label

CG ==> 0

OR ==> 1

filter only the column label and text\_

'''

df.replace(to\_replace = 'OR' , value ='1',inplace=True)

df.replace(to\_replace = 'CG' , value ='0',inplace=True)

df.shape

df=df[['label','text\_']]

df.sample(5)

'''

replace the column label

CG ==> 0

OR ==> 1

filter only the column label and text\_

'''

df.replace(to\_replace = 'OR' , value ='1',inplace=True)

df.replace(to\_replace = 'CG' , value ='0',inplace=True)

df.shape

df=df[['label','text\_']]

df.sample(5)

import nltk

def preprocess(reviews):

stop\_words = set(nltk.corpus.stopwords.words("english"))

tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+')

x=[]

for par in reviews:

tmp = []

sentences = nltk.sent\_tokenize(par)

for sent in sentences:

sent = sent.lower()

tokens = tokenizer.tokenize(sent)

filtered\_words = [w.strip() for w in tokens if w not in stop\_words and len(w) > 1]

tmp.extend(filtered\_words)

x.append(tmp)

return x

x = preprocess(df['text\_'])

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

vocab\_size = 0

def convert\_text\_to\_no(x):

global vocab\_size

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(x)

x=tokenizer.texts\_to\_sequences(x)

maxlen = 100

word\_index = tokenizer.word\_index

vocab\_size = len(tokenizer.word\_index) + 1

return pad\_sequences(x, maxlen=maxlen)

x=convert\_text\_to\_no(x)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Embedding, LSTM

#Defining Neural Network

model = Sequential()

#Non-trainable embeddidng layer

model.add(Embedding(

input\_dim=vocab\_size,

output\_dim=2,

input\_shape=(100,)

))

model.add(LSTM(units=128))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['acc'])

from sklearn.model\_selection import train\_test\_split

import numpy as np

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, df['label'].values)

X\_train = np.array(X\_train,dtype=np.float32)

X\_test = np.array(X\_test,dtype=np.float32)

y\_train = np.array(y\_train,dtype=np.float32)

y\_test = np.array(y\_test,dtype=np.float32)

y\_pred = (model.predict(X\_test) >= 0.5).astype("int")

from sklearn.metrics import accuracy\_score

accuracy\_score(y\_test, y\_pred)

model.save('fake\_review.h5')

from tensorflow.keras.models import load\_model

model = load\_model(r'/kaggle/working/fake\_review.h5')

review\_text = input("")

review\_text = preprocess([review\_text])

review\_text = convert\_text\_to\_no(review\_text)

review\_text = np.array(review\_text,dtype=np.float32)

result = list(model.predict(review\_text))[0]

if result >= 0.2:

print("Original Review")

else:

print("Fake Review")