Whose tweets?

Machine learning to identify people by their tweet content

Tools

Front-end

Html, JavaScript, D3

JavaScript Charts, Python,

Back-end

Flask server

Tweet API

Database

PostgreSQL

Amazon RDS server

Scikit-Learn

TensorFlow

Tokenizer

Deployed

Aws Amazon

http://nationallanguage.us-east-1.elasticbeanstalk.com/

Flowchart

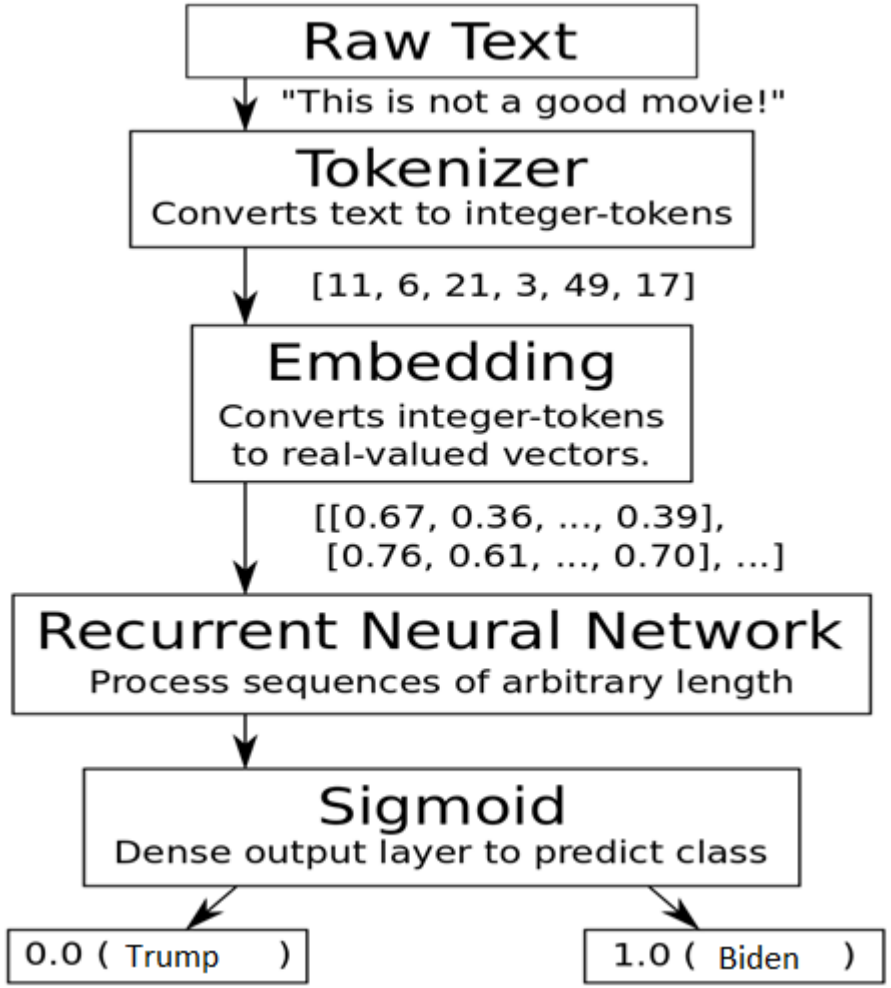
Procedure

Tokenize text-words into integer values

Embed integer-tokens real-valued vectors

Input embedding-vectors to a Recurrent Neural Network

Sigmoid-function output 0.0 and 1.0



Data Source

Load data

Tweet API

Retrieve tweets from Trump and Biden from 1/1/2020

Data clean

Removed all retweet from others , URL, @

Lower case tweet content

Remove tweets with less than 26 words

Database

Upload all data to PostgreSQL RDS server in Amazon

Trump: 4019 tweets

Biden: 3032 tweets

	date	name	tweet
0	2020-11-15	<JoeBiden>	congratulations to nasa and spacex on today's...
1	2020-11-14	<JoeBiden>	to the millions of hindus, jains, sikhs, and ...
2	2020-11-13	<JoeBiden>	i am the president-elect, but will not be pre...
3	2020-11-13	<JoeBiden>	i am alarmed by the surge in reported covid-1...
4	2020-11-13	<JoeBiden>	as the remnants of tropical storm eta continu...
...	...	...	...
3027	2020-01-02	<JoeBiden>	it was a privilege to work with during the o...
3028	2020-01-01	<JoeBiden>	every day that donald trump remains in the wh...
3029	2020-01-01	<JoeBiden>	this election is about the soul of our nation...
3030	2020-01-01	<JoeBiden>	with just over one month until the iowa caucu...
3031	2020-01-01	<JoeBiden>	every single human being deserves to be treat...

Classify, Tokenize, Padding, Truncating Data

Label

Trump tweet label 0, while Biden label 1

Split data (sklearn)

Train and Test Group

Tokenize

Tensorflow Tokenizer

Padding and Truncating

Tensorflow pad_sequences

Length: 57

Pre padding

```
x_train_tokens = tokenizer.texts_to_sequences(X_train)
x_test_tokens = tokenizer.texts_to_sequences(X_test)
```

```
print(X_train[418])
print(x_train_tokens[0])
```

```
 west virginia – in-person early voting is now open across the state.   make a plan to vote at
[867, 691, 25, 6, 329, 303, 287, 7, 47, 341, 183, 1, 117, 68, 5, 260, 2, 49, 36]
```

```
pad = 'pre'
x_train_pad = pad_sequences(x_train_tokens, maxlen=max_tokens,
                            padding=pad, truncating=pad)
x_test_pad = pad_sequences(x_test_tokens, maxlen=max_tokens,
                           padding=pad, truncating=pad)
```

```
x_test_pad
```

```
array([[   0,    0,    0, ..., 1195,  605,  110],
       [   0,    0,    0, ..., 1141,  520, 2136],
       [   0,    0,    0, ...,  432,    4,  118],
```

Recurrent Neural Network

Recurrent Neural Network (RNN)

Keras API

Embedding

Embedding-vector

Gated Recurrent Unit

Create 3 GRU layers

The output 16,8,4 as sequences outputs

Dense Layer

value between 0.0 and 1.0

as the classification output

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer_embedding (Embedding)  (None, 57, 8)             80000
_________________________________________________________________
gru_22 (GRU)                 (None, 57, 16)            1200
_________________________________________________________________
gru_23 (GRU)                 (None, 57, 8)             600
_________________________________________________________________
gru_24 (GRU)                 (None, 4)                 156
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 5
=================================================================
Total params: 81,961
Trainable params: 81,961
Non-trainable params: 0
```

Training and Evaluating

Training

validation_split=0.05

epochs=3, batch_size=64

```
%%time
model.fit(x_train_pad, y_train,
          validation_split=0.05, epochs=3, batch_size=64)
```

Evaluating

Loss: 0.305

Accuracy: 0.9169

```
%%time
# result = model.evaluate(x_test_pad, y_test)
model_loss, model_accuracy = model.evaluate(
    x_test_pad, y_test, verbose=2)
print(
    f"Normal Neural Network - Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
2143/2143 - 5s - loss: 0.3051 - accuracy: 0.9169
Normal Neural Network - Loss: 0.30507836667486177, Accuracy: 0.916938841342926
Wall time: 4.88 s
```

Testing sample data

Testing

Copy the latest tweet from Biden and Trump

Tokenized string and test result

```
# Biden  tweet
text1 = 'It’s not enough to praise our essential workers — we have to protect and pay them.'.lower()
text2 = 'The workers on the frontlines of this pandemic are making extraordinary sacrifices every single day. They
#trump tweet
text3 = 'Hope that all House Republicans will vote against Crazy Nancy Pelosiï War Powers Resolution'
text4 = 'PRESIDENTIAL HARASSMENT!'
text5 = 'IRAN WILL NEVER HAVE A NUCLEAR WEAPON!'
```

```
model.predict(tokens_pad)
```

```
array([[0.9928597 ],
       [0.9928119 ],
       [0.00724876],
       [0.01900795],
       [0.03351384],
       [0.03952959],
       [0.00930074],
       [0.00776094]], dtype=float32)
```

```
t should not be wasting their time and energy on a continuation of the totally
e as notification to the United States Congress that should Iran strike any U.S.
 text5, text6, text7, text8]
```

Save model and load model

Saving model

```
model.save("model1119-2.h5")
```

Load model

Load data in the flask server

```
model = tf.keras.models.load_model('model/model1119.h5’)
result = model.predict(tokens_pad)
```

Conclusion

- Natural Language Processing (NLP) using a Recurrent Neural Network with integer-tokens and an embedding layer. It works reasonably well if the hyper-parameters are chosen properly. But it is important to understand that this is not human-like comprehension of text. The system does not have any real understanding of the text. It is just a clever way of doing pattern-recognition.