Fake News Detection Model using TensorFlow in Python

Read | Discuss | Courses | Practice

Fake News means incorporating information that leads people to the wrong paths. It can have real-world adverse effects that aim to intentionally deceive, gain attention, manipulate public opinion, or damage reputation. It is necessary to detect fake news mainly for media outlets to have the ability to attract viewers to their website to generate online advertising revenue.

Fake News Detection Model using TensorFlow in Python

In this article, we are going to develop a Deep learning model using Tensorflow and use this model to detect whether the news is fake or not.

We will be using fake_news_dataset, which contains News text and corresponding label (FAKE or REAL). Dataset can be downloaded from this link.

The steps to be followed are :

1. Importing Libraries and dataset
2. Preprocessing Dataset
3. Generating Word Embeddings
4. Model Architecture
5. Model Evaluation and Prediction

Importing Libraries and Dataset

The libraries we will be using are :

- NumPy: To perform different mathematical functions.
- Pandas: To load dataset.
- Tensorflow: To preprocessing the data and to create the model.
- SkLearn: For train-test split and to import the modules for model evaluation.

Python3

```
import numpy as np
import pandas as pd
import json
import csv
import random

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import regularizers
```

```
from tensorflow.python.framework import ops
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
tf.disable_eager_execution()

# Reading the data
data = pd.read_csv("news.csv")
data.head()
```

Output :

	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
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

Preprocessing Dataset

As we can see the dataset contains one unnamed column. So we drop that column from the dataset.

Python3

```
data = data.drop(["Unnamed: 0"], axis=1)
data.head(5)
```

Output :

	title	text	label
0	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

Data Encoding

It converts the categorical column (label in out case) into numerical values.

Python3

```
# encoding the labels
le = preprocessing.LabelEncoder()
le.fit(data['label'])
data['label'] = le.transform(data['label'])
```

These are some variables required for the model training.

Python3

```
embedding_dim = 50
max_length = 54
trunc_type = 'post'
padding_type = 'post'
oov_tok = "<OOV>"
training_size = 3000
test_portion = .1
```

This process divides a large piece of continuous text into distinct units or tokens basically. Here we use columns separately for a temporal basis as a pipeline just for good accuracy.

Python3

```
title = []
text = []
labels = []
for x in range(training_size):
    title.append(data['title'][x])
    text.append(data['text'][x])
    labels.append(data['label'][x])
```

Applying Tokenization

Python3

```
tokenizer1 = Tokenizer()
tokenizer1.fit_on_texts(title)
word_index1 = tokenizer1.word_index
vocab_size1 = len(word_index1)
sequences1 = tokenizer1.texts_to_sequences(title)
padded1 = pad_sequences(
    sequences1,  padding=padding_type, truncating=trunc_type)
split = int(test_portion * training_size)
training_sequences1 = padded1[split:training_size]
test_sequences1 = padded1[0:split]
test_labels = labels[0:split]
training_labels = labels[split:training_size]
```

Generating Word Embedding

It allows words with similar meanings to have a similar representation. Here each individual word is represented as real-valued vectors in a predefined vector space. For that we will use **glove.6B.50d.txt.** It has the predefined vector space for words. You can download the file using this link.

Python3

```
embeddings_index = {}
with open('glove.6B.50d.txt') as f:
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs

# Generating embeddings
embeddings_matrix = np.zeros((vocab_size1+1, embedding_dim))
for word, i in word_index1.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embeddings_matrix[i] = embedding_vector
```

Creating Model Architecture

Now it's time to introduce TensorFlow to create the model. Here we use the TensorFlow embedding technique with Keras Embedding Layer where we map original input data into some set of real-valued dimensions.

Python3

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size1+1, embedding_dim,
```

```
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Conv1D(64, 5, activation='relu'),
        tf.keras.layers.MaxPooling1D(pool_size=4),
        tf.keras.layers.LSTM(64),
        tf.keras.layers.Dense(1, activation='sigmoid')
    ])
    model.compile(loss='binary_crossentropy',
                  optimizer='adam', metrics=['accuracy'])
    model.summary()
```

Output :

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding)      (None, 54, 50)            377600

 dropout_1 (Dropout)          (None, 54, 50)            0

 conv1d_1 (Conv1D)            (None, 50, 64)            16064

 max_pooling1d_1 (MaxPooling  (None, 12, 64)           0
 1D)

 lstm_1 (LSTM)                (None, 64)                33024

 dense_1 (Dense)              (None, 1)                 65

=================================================================
Total params: 426,753
Trainable params: 49,153
Non-trainable params: 377,600
_________________________________________________________________
```

Python3

```
num_epochs = 50
```

```
testing_padded = np.array(test_sequences1)
testing_labels = np.array(test_labels)

history = model.fit(training_padded, training_labels,
                    epochs=num_epochs,
                    validation_data=(testing_padded,
                                     testing_labels),
                    verbose=2)
```

Output :

```
Epoch 44/50
2700/2700 - 2s - loss: 0.0695 - accuracy: 0.9707 - val_loss: 0.9628 - val_accuracy: 0.7600 - 2s/epoch - 785us/sample
Epoch 45/50
2700/2700 - 2s - loss: 0.0509 - accuracy: 0.9826 - val_loss: 1.0151 - val_accuracy: 0.7467 - 2s/epoch - 786us/sample
Epoch 46/50
2700/2700 - 2s - loss: 0.0635 - accuracy: 0.9752 - val_loss: 1.0047 - val_accuracy: 0.7700 - 2s/epoch - 798us/sample
Epoch 47/50
2700/2700 - 2s - loss: 0.0600 - accuracy: 0.9789 - val_loss: 0.9136 - val_accuracy: 0.7367 - 2s/epoch - 786us/sample
Epoch 48/50
2700/2700 - 2s - loss: 0.0658 - accuracy: 0.9774 - val_loss: 0.9439 - val_accuracy: 0.7500 - 2s/epoch - 811us/sample
Epoch 49/50
2700/2700 - 2s - loss: 0.0557 - accuracy: 0.9815 - val_loss: 0.9449 - val_accuracy: 0.7700 - 2s/epoch - 788us/sample
Epoch 50/50
2700/2700 - 2s - loss: 0.0691 - accuracy: 0.9741 - val_loss: 0.9306 - val_accuracy: 0.7500 - 2s/epoch - 782us/sample
```


Model Evaluation and Prediction

Now, the detection model is built using TensorFlow. Now we will try to test the model by using some news text by predicting whether it is true or false.

Python3

```
# sample text to check if fake or not
X = "Karry to go to France in gesture of sympathy"
```

```
    sequences = pad_sequences([sequences], maxlen=54,
                              padding=padding_type,
                              truncating=trunc_type)
    if(model.predict(sequences, verbose=0)[0][0] >= 0.5):
        print("This news is True")
    else:
        print("This news is false")
```

Output :

```
This news is false
```

Conclusion

In this way, we can build a fake news detection model using TensorFlow using python.

Last Updated : 09 Oct, 2022

Previous

Predicting Stock Price Direction using Support Vector Machines

Next

CIFAR-10 Image Classification in TensorFlow