

# Text Classification Model on Stack Overflow

Keqi Yu

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

# Problem Statement



**Stack Overflow** is the largest online community for programmers to learn, share their knowledge, and advance their careers.

## Predicting the tags of questions from Stack Overflow

Focus on the questions containing 5 possible ML-related tags:

**Tensorflow|keras|matplotlib  
pandas|scikit-learn**

# Data Used

<https://console.cloud.google.com/marketplace/product/stack-exchange/stack-overflow?project=just-nova-382506>

```
SELECT tags,
REGEXP_REPLACE(text,
r"(?:tensorflow|tf|keras|matplotlib|plt|pd|sklearn|pandas|scikit-learn)",
"apple") as text
FROM `bigquery-public-data.stackoverflow.posts_questions`
WHERE REGEXP_CONTAINS(tags, r"(?:tensorflow|keras|matplotlib|pandas|scikit-learn)") as results;
```

Public dataset **stackoverflow** from **Google BigQuery**, includes an archive of Stack Overflow content, including posts, votes, tags, and badges.

index	original_tags	text	tags
62312	python,tensorflow,keras	apple - concatenating two inputs of same 1st and 3rd dimension but differing 2nd dimension i'm creating a functional api apple model to regress and predict a numerical value based on two mixed-data inputs. the network consists of two models, the outputs of which are intended to be concatenated and input into another model, which will output the final value to be compared against the value to predict. my (unfinished) code looks like this: def categorical_model(): inputa = input(shape=(3, 1329, )) x = dense(8, activation='relu')(inputa) x = dense(4, activation='relu')(x) return model(inputs=inputa, outputs=x) def continuous_model(): inputb = input(shape=(1329, 2)) y = dense(64, activation='relu')(inputb) y = dense(32, activation='relu')(y) y = dense(4, activation='relu')(y) return model(inputs=inputb, outputs=y) cat = categorical_model() con = continuous_model() catcon_list = [cat.output, con.output] concatenated = concatenate(catcon_list, axis=0, name = 'concatenate') concatenated = dense(2, activation='softmax')(concatenated) merged = model(input=[cat.input, con.input], output=concatenated) merged.summary() the code results in a valueerror as such: valueerror: a 'concatenate' layer requires inputs with matching shapes except for the concat axis. got inputs shapes: [(none, 3, 4), (none, 1329, 4)] can these inputs be concatenated? how can i change the axis to the 2nd dimension?	tensorflow,keras
75401	python,pandas	find first occurrence and get column name in columns apple dataframe i am trying to get first occurrence in months and get the column name as a new column in a apple dataframe here is the df; df = apple.dataframe({'jan':[0,0,5], 'feb':[0,0,0], 'mar':[1,0,0], 'apr':[8,2,0], 'may':[0,4,10], 'june':[3,2,3]}) result should be mar,apr,jan accordingly i know that i can make it via for loop iterating one by one but seeking for much elegant solution.	pandas
404914	python,csv,pandas,blank-line	how not to choose blank rows in python? i wrote the following code in python to choose only selected rows. however 'activity_url.csv' has blank rows. hence it is giving me an error. so how do i skip the blank rows? data = apple.read_csv('activity_url.csv', delimiter=';') x="http" url_data=np.array(data[data.iloc[:,1].str.contains(x, na=False)]);[1]	pandas

# Data Preprocessing - 439918 rows $\times$ 2 columns

- We'll use an 80/20 train/test split
- Create our Keras Tokenizer object
- Bag of words is one approach to converting free-form text input into matrices.
- Encoding tags as multi-hot arrays using Scikit-learn's MultiLabelBinarizer

```
from tensorflow.keras.preprocessing import text

class TextPreprocessor(object):
    def __init__(self, vocab_size):
        self._vocab_size = vocab_size
        self._tokenizer = None

    def create_tokenizer(self, text_list):
        tokenizer = text.Tokenizer(num_words=self._vocab_size)
        tokenizer.fit_on_texts(text_list)
        self._tokenizer = tokenizer

    def transform_text(self, text_list):
        text_matrix = self._tokenizer.texts_to_matrix(text_list)
        return text_matrix
```

```
tag_encoder = MultiLabelBinarizer()
tags_encoded = tag_encoder.fit_transform(tags_split)
num_tags = len(tags_encoded[0])
print(data['tags'].values[0])
print(tag_encoder.classes_)
print(tags_encoded[0])

tensorflow,keras
['keras' 'matplotlib' 'pandas' 'scikit-learn' 'tensorflow']
[1 0 0 0 1]
```

[illegible]

# Algorithm

I used the **Keras Sequential Model**.

The first layer takes our 400-element vocabulary vector as input and transforms it into a 50 neuron layer.

Then it takes this 50-neuron layer and transforms it into a 25-neuron layer.

The size of our last layer will be equivalent to the number of tags in our dataset.

Sigmoid will convert each of our 5 outputs to a value between 0 and 1 indicating the probability that a specific label corresponds with that input.

```
[34] def create_model(vocab_size, num_tags):  
  
    model = tf.keras.models.Sequential()  
    model.add(tf.keras.layers.Dense(50, input_shape=(VOCAB_SIZE,), activation='relu'))  
    model.add(tf.keras.layers.Dense(25, activation='relu'))  
    model.add(tf.keras.layers.Dense(num_tags, activation='sigmoid'))  
  
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])  
    return model  
  
[35] model = create_model(VOCAB_SIZE, num_tags)  
    model.summary()  
  
# Train and evaluate the model  
model.fit(body_train, train_tags, epochs=3, batch_size=128, validation_split=0.1)  
print('Eval loss/accuracy:{}'.format(  
    model.evaluate(body_test, test_tags, batch_size=128)))  
  
# Export the model to a file  
model.save('keras_saved_model.h5')
```

# Result

Accuracy: 87.13%

```
[74] test_requests = [  
    "Tensor('args_0:0', shape=(), dtype=string), type: <class 'tensorflow.python.framework.ops.Tensor'>, valid types: <clas  
    "I am looking for a way to resample 5min data to 4hr data with the same 4hr intervals ie. 0:00 , 4:00,8:00 regardless o  
    ]
```

```
from model_prediction import CustomModelPrediction  
  
classifier = CustomModelPrediction.from_path('.')  
results = classifier.predict(test_requests)  
print(results)  
  
for i in range(len(results)):  
    print('Predicted labels:')  
    for idx, val in enumerate(results[i]):  
        if val > 0.7:  
            print(tag_encoder.classes_[idx])  
    print('\n')
```

```
1/1 [=====] - 0s 53ms/step  
[[0.23234456777572632, 0.0019265312002971768, 0.004992923233658075, 0.007039365358650684, 0.9  
Predicted labels:  
tensorflow
```

```
Predicted labels:  
pandas
```

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 50)	20050
dense_4 (Dense)	(None, 25)	1275
dense_5 (Dense)	(None, 5)	130

=====  
Total params: 21,455  
Trainable params: 21,455  
Non-trainable params: 0

```
Epoch 1/3  
2475/2475 [=====] - 9s 3ms/step - loss: 0.1319 - accuracy: 0.8528 - val_loss: 0.1084 - val_accuracy: 0.8806  
Epoch 2/3  
2475/2475 [=====] - 7s 3ms/step - loss: 0.1065 - accuracy: 0.8755 - val_loss: 0.1057 - val_accuracy: 0.8682  
Epoch 3/3  
2475/2475 [=====] - 7s 3ms/step - loss: 0.1028 - accuracy: 0.8778 - val_loss: 0.1038 - val_accuracy: 0.8734  
688/688 [=====] - 1s 2ms/step - loss: 0.1050 - accuracy: 0.8713  
Eval loss/accuracy:[0.10499855875968933, 0.8713402152061462]
```

