



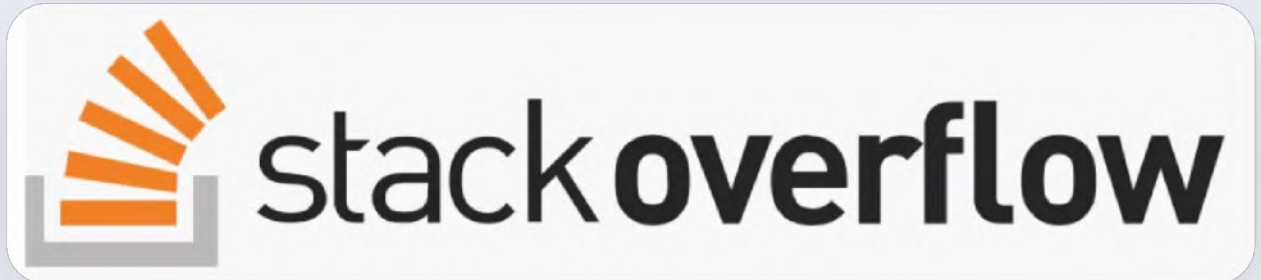
COURSE: MET CS 664 ARTIFICIAL INTELLIGENCE
BOSTON UNIVERSITY METROPOLITAN COLLEGE

DATE: DECEMBER 18TH 2024

Artificial Intelligence Final Project



Course Instructor: Suresh Kalathur



Predicting Closed Questions on Stack Overflow

USING NEURAL NETWORK AND NATURAL LANGUAGE PROCESSING MODELS -
(MACHINE LEARNING + ARTIFICIAL INTELLIGENCE)

Meet the team



Shreni Singh
TEAM MEMBER 1



Srujana Niranjankumar
TEAM MEMBER 2

Stack Overflow is an *invaluable resource for programmers*, serving as a global forum for coding questions and answers. However, not all questions meet the *platform's quality and relevance standards*. Questions may be closed for reasons such as duplication, lack of detail, opinion-based content, or being off-topic.

Identifying such questions at the time of submission can help improve the platform's efficiency and user experience.

This project aims to build an AI-driven predictive system that determines whether a newly posted question on Stack Overflow will be closed, as well as identifying the likely reason for closure. By addressing this issue, we can contribute to enhancing the community's effectiveness and the quality of knowledge shared.

Issue with BERT Preprocessor model in TF2 and python

Asked 9 months ago Modified 4 months ago Viewed 932 times Part of NLP Collective



I am trying to use BERT to do a text classification project. However I keep running into this error

```
ValueError                                Traceback (most recent call last)
Cell In[37], line 4
      2 text_input = tf.keras.Input(shape=(), dtype=tf.string, name='text')
      3 bert_preprocess = hub.KerasLayer(preprocess_url, name='preprocessing')
----> 4 preprocessed_text = bert_preprocess(text_input)
      5 bert_encoder = hub.KerasLayer(encoder_url,
      6                               trainable=True,
      7                               name='BERT_encoder')
      8 outputs = bert_encoder(preprocessed_text)

ValueError: Exception encountered when calling layer 'preprocessing' (type Keras
A KerasTensor is symbolic: it's a placeholder for a shape and a dtype. It doesn't

Call arguments received by layer 'preprocessing' (type KerasLayer):
  • inputs=<KerasTensor shape=(None,), dtype=string, sparse=None, name=text>
  • training=None

A KerasTensor is symbolic: it's a placeholder for a shape and a dtype. It doesn't
```

when building this model:

What should I do if a question is closed before I finish my answer? [duplicate]

Asked 8 years, 9 months ago Modified 8 years, 9 months ago Viewed 204 times

▲

-10

▼

🔖

🕒

This question already has answers here:
Questions being closed as answers are being written (5 answers)
Closed 8 years ago.

The Overflow Blog

Legal advice from a

Hot Meta Posts

11 Upcoming voting ex encourage people v vote

Today, I was in the middle of answering a question, but was interrupted by a notification that it's closed (off-topic, but it was about programming, despite the fact it was poorly asked* it requested something that could be done in numerous ways), and no more answers can be posted. The question may be able to be salvaged.

What should I do if I have already started an answer (that I felt was succinct and would be helpful to OP and maybe others) and it closes? It's a problem that happens often it seems.

I didn't want to trash my answer, but I didn't feel like fighting to reopen the question (at least right now). Personally, I felt the close was just too fast, as the OP may have first posted the question and then gone to get some code to add or something.

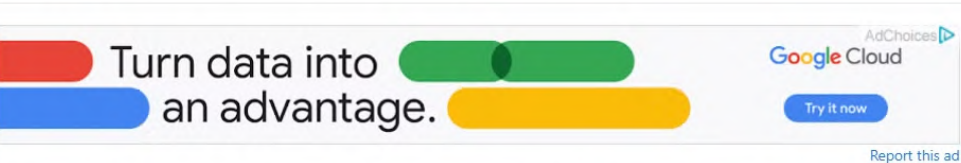
Should answers be allowed to be submitted, as long as the question is less than 5 or 10 minutes old, even if they are submitted after it closes? Of course, if this was done, I would keep the alert for closed question, and also add a disclaimer that you are answering a closed question and your answer probably won't be around unless you fix the question. That way, the OP has an answer, at least, and if the question is fixed and reopened, people don't need to redo their work.

Could we allow answers to be "submitted" to closed questions in such a way that they are only visible to the author? If the question reopens, you should get a notification and after editing your answer, it would become visible to all.

By "poor quality" I meant they had misspellings and other English problems. It was very obvious what they meant, however.

CNN architecture

Asked 7 years, 7 months ago Modified 4 years, 1 month ago Viewed 2k times



▲

1

▼

🔖

🕒

At first, this question is less about programming itself but about some logic behind the CNN architecture. I do understand how every layer works but my only question is: Does it make sense to separate the ReLU and Convolution-Layer? I mean, can a ConvLayer exist and work and update its weights by using backpropagation without having a ReLU behind it?

I thought so. This is why I created the following independent layers:

1. ConvLayer
2. ReLU
3. Fully Connected
4. Pooling
5. Transformation (transform the 3D output into one dimension) for ConvLayer -> Fully Connected.

I am thinking about merging Layer 1 and 2 into one. What should I go for?

conv-neural-network convolution

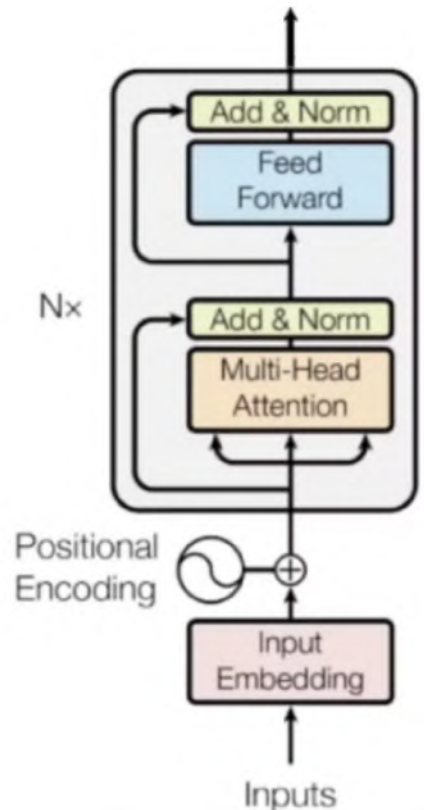
Problem Statement

The increasing volume of questions on Stack Overflow necessitates a system to automatically flag questions that are likely to be closed. This project focuses on:

- Predicting whether a question will be closed.
- Classifying the closure reason (e.g., not a real question, off-topic, too localized).

Dataset and Approach

- **Dataset:** The Kaggle dataset comprises a rich set of features derived from questions posted on Stack Overflow, including:
 - **Question Title:** A short description of the question.
 - **Question Body:** Detailed text of the question.
 - **Tags:** Metadata describing topics related to the question.
 - **Post Creation Date:** Date and time the question was created.
 - **Closure Reason (or Open Status):** Labels such as "duplicate," "off-topic," etc.
 - No.of data point ~ 15 lakh
- **Approach:**
 - Use a **BERT, FNN, DNN model** for text classification.
 - Fine-tune BERT for predicting Stack Overflow questions' status.
 - Use **Basic Neural Network** and **Deep Neural Network** for predicting Stack Overflow questions status.



Column Name	Description
PostId	Unique identifier for the post
PostCreationDate	Date when the post was created
OwnerUserId	ID of the user who posted it
OwnerCreationDate	Date the user account was created
ReputationAtPostCreation	User's reputation at post creation
OwnerUndeletedAnswerCountAtPostTime	Number of answers by the user at that time
Title	Title of the post
BodyMarkdown	Content of the post
Tag1, Tag2, Tag3, Tag4, Tag5	Tags assigned to the post
PostClosedDate	Date the post was closed
OpenStatus	Indicates if the post is open or closed

Data Preprocessing

- **Preprocessing Steps:**
 - Combining multiple text columns(e.g., Title, BodyMarkdown) into a single text field
 - Label encoding of the OpenStatus categories to numerical values.
 - Transforming the text data into numerical features using TF-IDF vectorizer
 - Tokenize the question text using BERT's preprocessing layer.
 - Handle missing values, drop irrelevant columns.
 - Encode target labels using custom mapping.

```
# Define the columns to be
removed
columns_to_remove = ['Title',
'BodyMarkdown']
# Drop the specified
columns from each
DataFrameso_train_df.drop(
columns=columns_to_remov
e, inplace = True)
```

46%

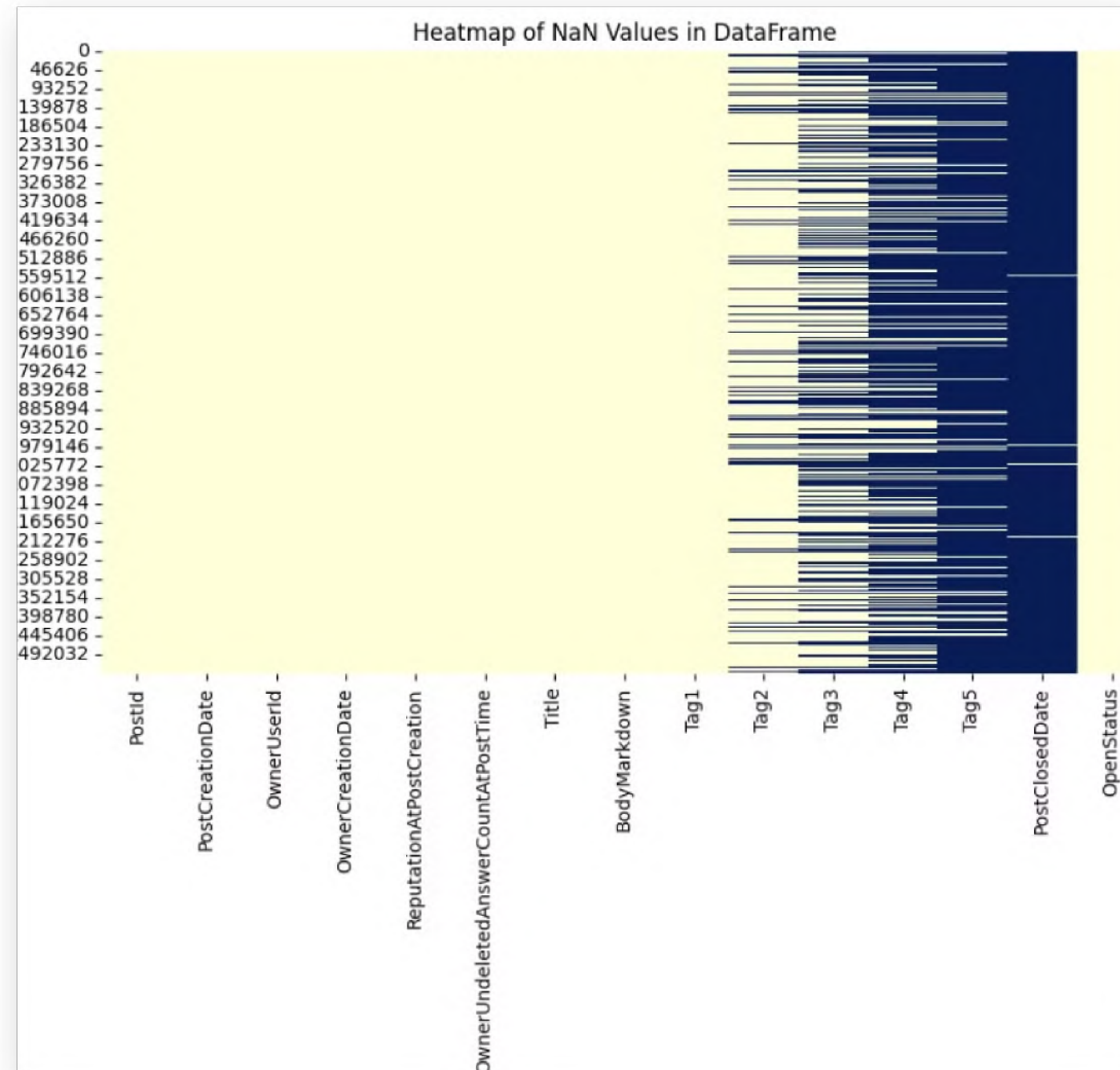
```
# Tokenize input text using
BERT preprocessing layer
preprocessor =
hub.KerasLayer("https://kaggl
e.com/models/tensorflow/be
rt/frameworks/TensorFlow2/v
ariations/en-uncased-
preprocess/versions/3")
tokenized_output =
preprocessor(input_text)
```

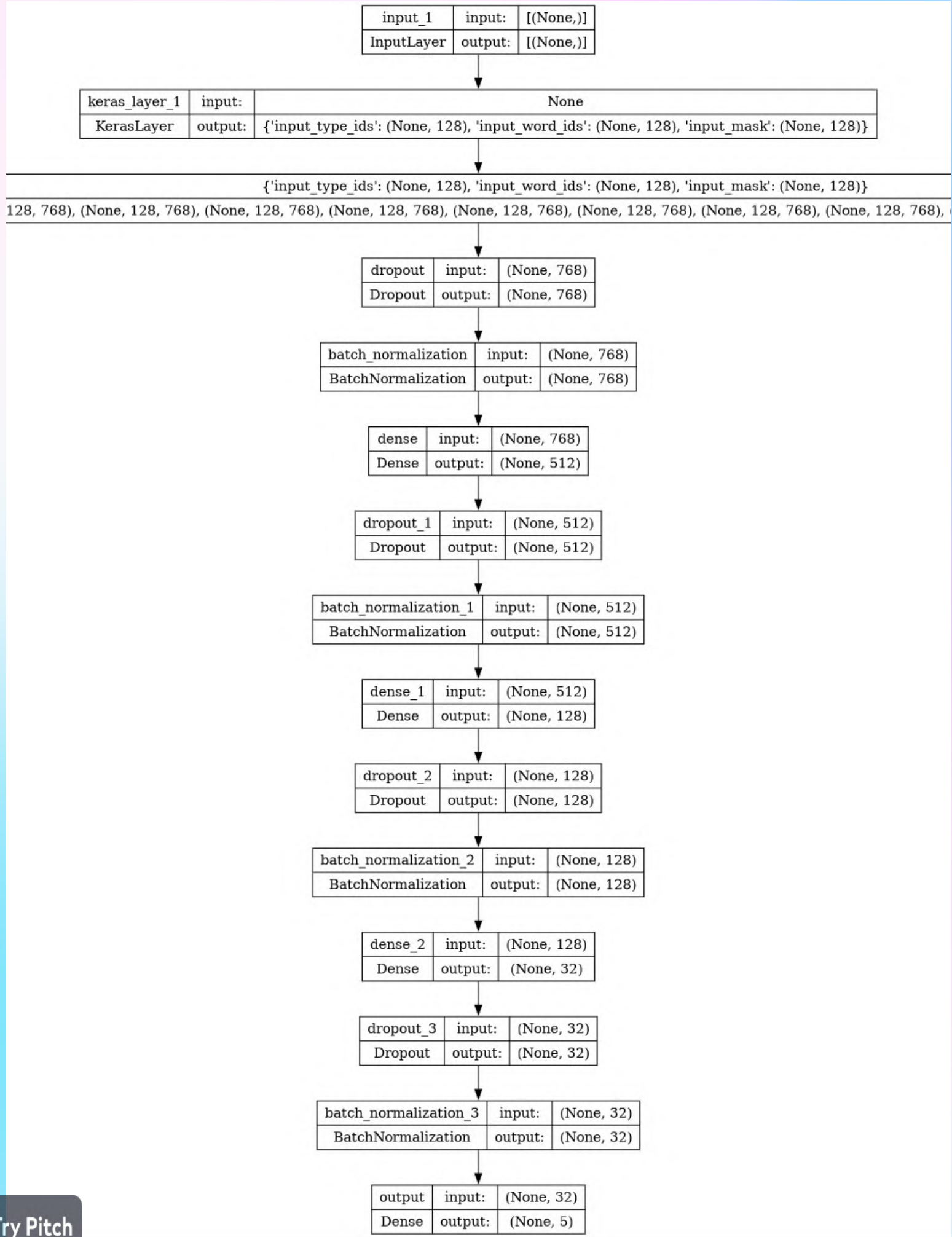
46%

Data Preprocessing

HEATMAP

We have decided to utilize only the "Title," "BodyMarkdown," and "OpenStatus" columns from the DataFrame. The "Title" and "BodyMarkdown" columns will be combined to form the text input, while the "OpenStatus" column will serve as the target variable.





Model Architecture

- **Model Overview:**
 - **BERT encoder** layer to understand contextual relationships in the text.
 - **Dense layers** with dropout and batch normalization for classification.
- **Key Layers:**
 - BERT encoder layer.
 - Dense hidden layers (512, 128, 32 units).
 - Output layer with **softmax** activation (5 classes for Stack Overflow question types).

Component	Type	Description
Text Processing	BERT Preprocessor	Tokenization and text processing
Main Model	BERT Encoder	Transformer-based model (not RNN, LSTM, CNN)
Classification Head	Feedforward Neural Network (MLP)	Custom dense layers for classification

Model Architecture

- **Summary of NN Architecture:**
 - **Input Layer:** 10000 features (from TF-IDF).
 - **Hidden Layer 1:** 128 neurons, ReLU activation.
 - **Dropout Layer:** 50% dropout rate.
 - **Hidden Layer 2:** 64 neurons, ReLU activation.
 - **Output Layer:** 5 neurons, Softmax activation.
 -
- **Summary of DNN Architecture:**
 - **Input Layer:** 10000 features (from TF-IDF).
 - **Hidden Layer 1:** 512 neurons, ReLU activation.
 - **Hidden Layer 2:** 256 neurons, ReLU activation.
 - **Hidden Layer 3:** 128 neurons, ReLU activation.
 - **Dropout Layer:** 50% dropout rate.
 - **Output Layer:** 5 neurons, Softmax activation.

```
# TF-IDF Vectorizer for text processing
vectorizer = TfidfVectorizer(max_features=10000)

# Fit and transform the text data
X_train_tfidf = vectorizer.fit_transform(X_train)
X_valid_tfidf = vectorizer.transform(X_valid)
X_test_tfidf = vectorizer.transform(X_test)

# Save the vectorizer for later use
with open('vectorizer.pkl', 'wb') as f:
    pickle.dump(vectorizer, f)

# Label Encoder for OpenStatus encoding
encoder = LabelEncoder()
y_train_encoded = encoder.fit_transform(y_train)
y_valid_encoded = encoder.transform(y_valid)
y_test_encoded = encoder.transform(y_test)

# Save the encoder for later use
with open('encoder.pkl', 'wb') as f:
    pickle.dump(encoder, f)
```

```
def create_nn_model(input_dim):
    model = Sequential()
    model.add(Dense(128, input_dim=input_dim, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(5, activation='softmax')) # 5 classes in OpenStatus
    model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

nn_model = create_nn_model(X_train_tfidf.shape[1])
nn_model.summary()

# Train the model
nn_model.fit(X_train_tfidf, y_train_encoded, validation_data=(X_valid_tfidf, y_valid_encoded), epochs=5, batch_size=32)

# Save the model
nn_model.save('nn_model.h5')
```


Model Training

- **Training Strategy:**
 - **Optimizer:** Adam with a learning rate of 2e-5.
 - **Loss Function:** Sparse categorical cross-entropy.
 - **Metrics:** Accuracy

```
# Classification layers
# Add dropout and batch normalization layers
drop1 = tf.keras.layers.Dropout(0.5)(pooled_output)
batch_norm1 = tf.keras.layers.BatchNormalization()(drop1)

# Add hidden dense layers
hidden1 = tf.keras.layers.Dense(512, activation='relu')(batch_norm1)
drop2 = tf.keras.layers.Dropout(0.4)(hidden1)
batch_norm2 = tf.keras.layers.BatchNormalization()(drop2)

hidden2 = tf.keras.layers.Dense(128, activation='relu')(batch_norm2)
drop3 = tf.keras.layers.Dropout(0.3)(hidden2)
batch_norm3 = tf.keras.layers.BatchNormalization()(drop3)

hidden3 = tf.keras.layers.Dense(32, activation='relu')(batch_norm3)
drop4 = tf.keras.layers.Dropout(0.2)(hidden3)
batch_norm4 = tf.keras.layers.BatchNormalization()(drop4)

# Output layer with 5 classes
output_layer = tf.keras.layers.Dense(5, activation='softmax', name='output')(batch_norm4)

# Model definition
model = tf.keras.Model(inputs=[text_input], outputs=[output_layer])

# Model Compilation
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=2e-5), # Adam optimizer with a small
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False), # Sparse cate
              metrics=['accuracy']) # Tracking accuracy
```

Model: "model"			
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None,)]	0	[]
keras_layer_1 (KerasLayer)	{'input_type_ids': (None, 128), 'input_word_ids': (None, 128), 'input_mask': (None, 128)} }	0	['input_1[0][0]']
keras_layer_2 (KerasLayer)	{'encoder_outputs': [(None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768), (None, 128, 768)], 'pooled_output': (None, 768), 'sequence_output': (None, 128, 768), 'default': (None, 768)} }	109482241	['keras_layer_1[0][0]', 'keras_layer_1[0][1]', 'keras_layer_1[0][2]']
dropout (Dropout)	(None, 768)	0	['keras_layer_2[0][13]']
batch_normalization (Batch Normalization)	(None, 768)	3072	['dropout[0][0]']
dense (Dense)	(None, 512)	393728	['batch_normalization[0][0]']
dropout_1 (Dropout)	(None, 512)	0	['dense[0][0]']
batch_normalization_1 (Batch Normalization)	(None, 512)	2048	['dropout_1[0][0]']
dense_1 (Dense)	(None, 128)	65664	['batch_normalization_1[0][0]']
dropout_2 (Dropout)	(None, 128)	0	['dense_1[0][0]']
batch_normalization_2 (Batch Normalization)	(None, 128)	512	['dropout_2[0][0]']
dense_2 (Dense)	(None, 32)	4128	['batch_normalization_2[0][0]']
dropout_3 (Dropout)	(None, 32)	0	['dense_2[0][0]']
batch_normalization_3 (Batch Normalization)	(None, 32)	128	['dropout_3[0][0]']
output (Dense)	(None, 5)	165	['batch_normalization_3[0][0]']
=====			
Total params: 109951686 (419.43 MB)			
Trainable params: 109948805 (419.42 MB)			
Non-trainable params: 2881 (11.25 KB)			

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	1,280,128
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 5)	325

Total params: 1,288,709 (4.92 MB)

Trainable params: 1,288,709 (4.92 MB)

Non-trainable params: 0 (0.00 B)

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 512)	5,120,512
dropout_1 (Dropout)	(None, 512)	0
dense_4 (Dense)	(None, 256)	131,328
dense_5 (Dense)	(None, 128)	32,896
dense_6 (Dense)	(None, 5)	645

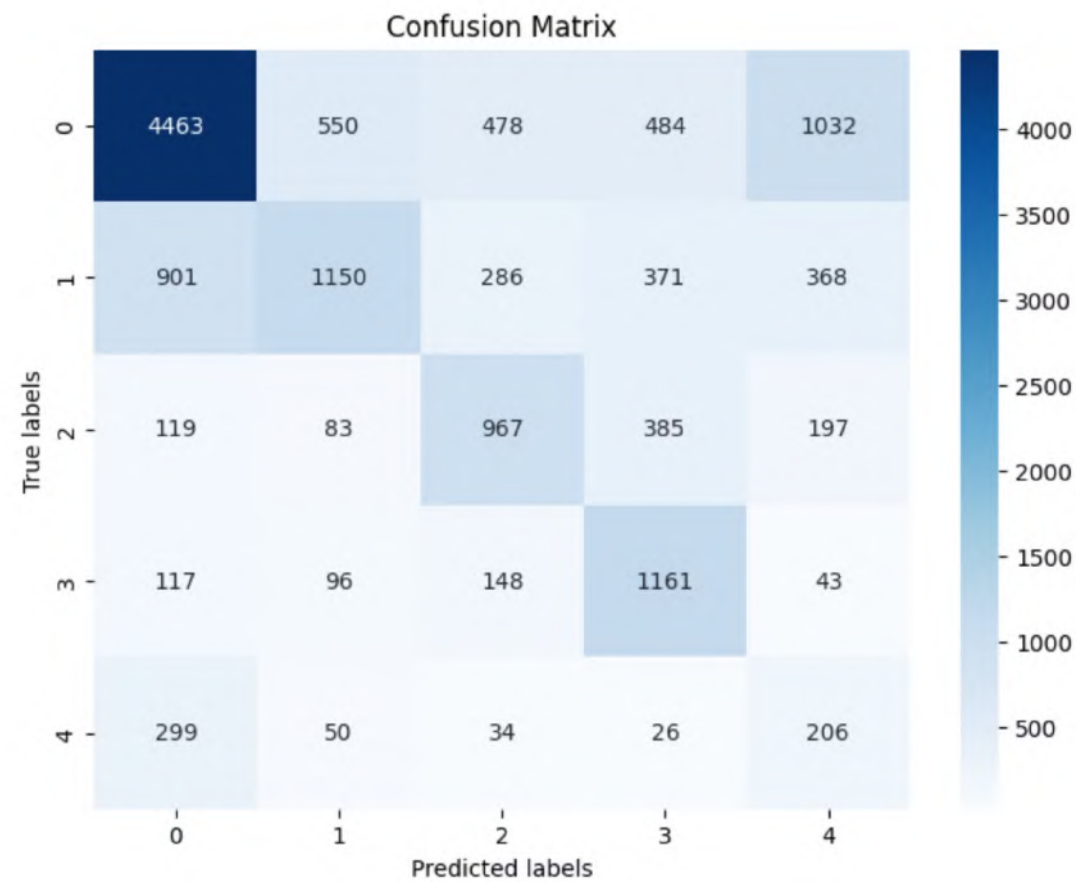
Total params: 5,285,381 (20.16 MB)

Trainable params: 5,285,381 (20.16 MB)

Non-trainable params: 0 (0.00 B)

Model Training

- **Model Creation:**
 - Two different types of neural network models are created:
Feedforward Neural Network (NN) and **Deep Neural Network (DNN)**.
 - Both models use the **ReLU activation function** for hidden layers and **softmax activation** for the output layer (5 classes).
 - The models are trained using **sparse categorical crossentropy** loss and **Adam optimizer**.
 - After training, the models are saved as .h5 files for later use.
- **Training and Saving Models:**
 - The **NN model** is trained for 5 epochs with a batch size of 32.
 - Similarly, the **DNN model** is trained using the same parameters.
 - Both models are saved for future use.



141/141 3s 16ms/step

accuracy: 0.9364 - loss: 0.2564 - val_accuracy: 0.4875 - val_loss: 1.6017

141/141 7s 52ms/step

accuracy: 0.9926 - loss: 0.0305 - val_accuracy: 0.4589 - val_loss: 2.6909

438/438 [=====] - 91s 206ms/step

Accuracy: 56.7076 %

Sparse Categorical Cross-Entropy Loss: 1.1826568035936256

Model Evaluation & Results

- **Performance Metrics:**
 - **Accuracy:** Measures the overall correctness of the model.
 - **Confusion Matrix:** Helps to understand misclassifications and the model's performance for each class.
- **Sample Output:**
 - **Accuracy:** 57-61%
 - Confusion matrix highlighting prediction correctness across classes.

TEXT:

Title: 'Hardware. What is the difference between a port and a bank?'

For example, NVidia's shared memory is 32-banked, so what they say is then what is port ? also same issues to cache structure

Could anyone clarify on this ?

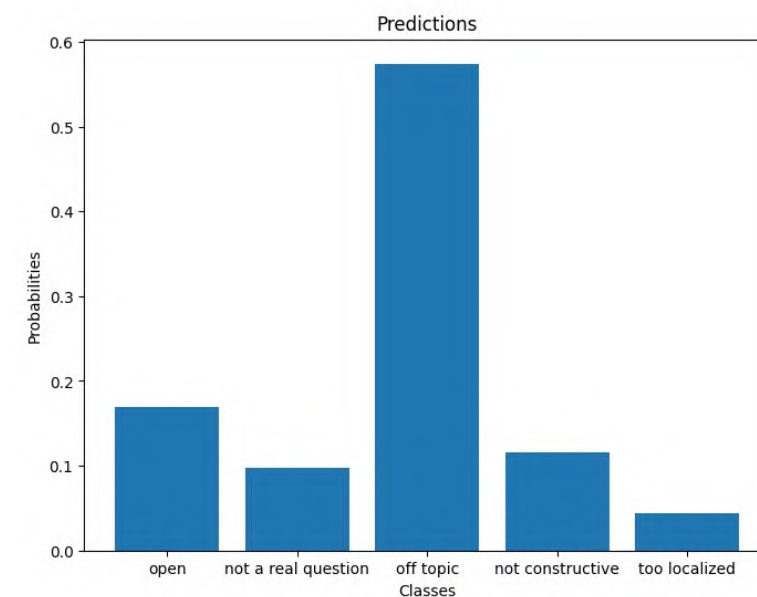
Thanks!

GROUND TRUTH:

off topic

1/1 [=====] - 0s 38ms/step

PREDICTION:



PREDICTED CLASS: off topic

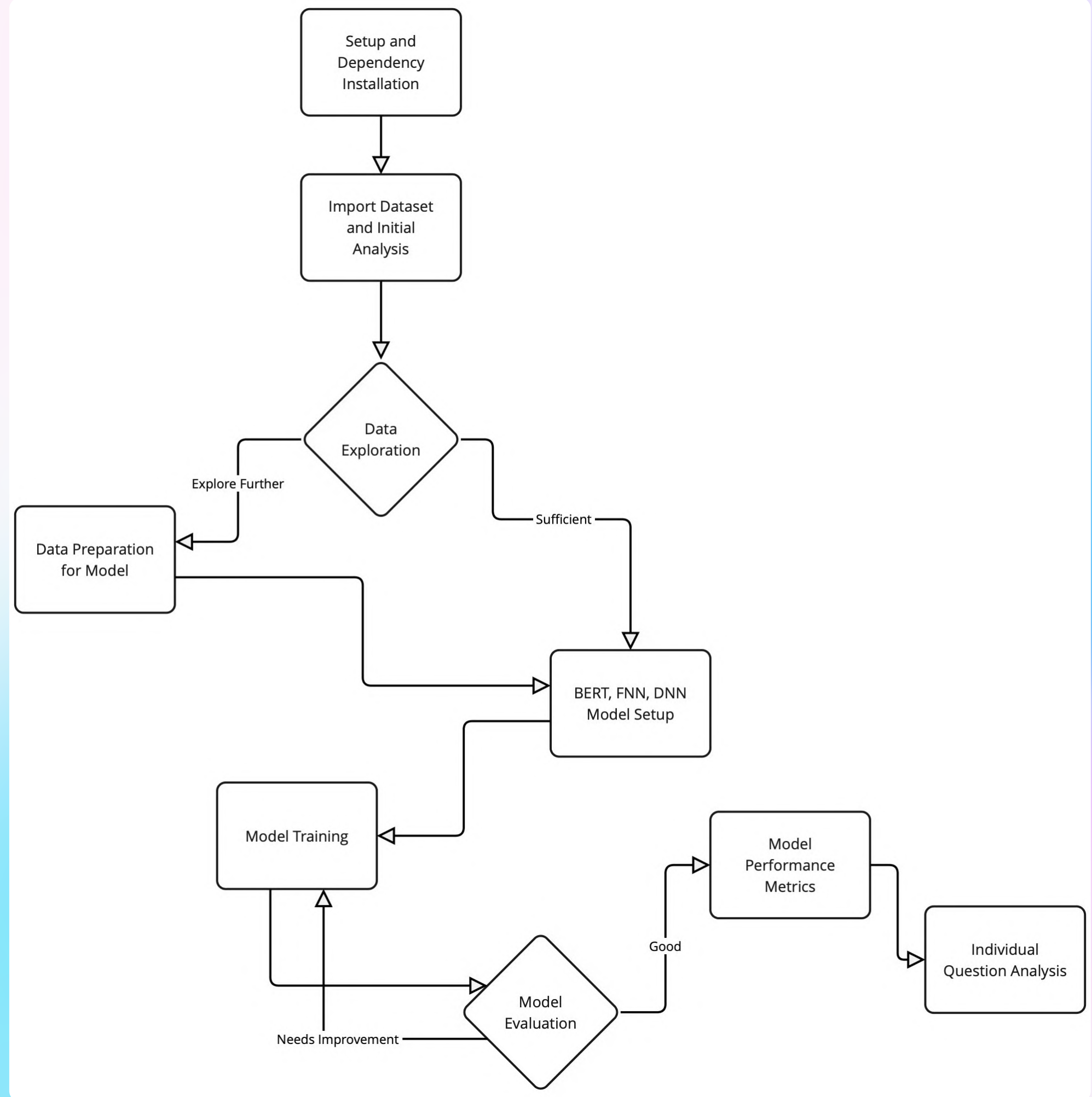
CORRECT PREDICTION !!!

Prediction Demonstration

- **Random Test Case:**
 - **Input Question:** Example of a Stack Overflow question.
 - **Predicted Class:** The model's predicted class (closed, off-topic, etc.).
 - **Ground Truth:** Actual label for the question.
- **Visualization:** Bar chart showing the prediction probabilities for each class.



Flowchart



Challenges and Future Work

- **Challenges:**
 - Handling unbalanced classes and making predictions on noisy text.
 - Fine-tuning BERT effectively to achieve optimal results.
- **Future Work:**
 - Experimenting with other models like **RoBERTa** or **DistilBERT**.
 - Incorporating more features (e.g., user reputation, question history).

Conclusion

By leveraging **neural networks** and **advanced NLP techniques**, this project seeks to assist Stack Overflow users and moderators in identifying and addressing low-quality or inappropriate questions promptly.

The anticipated system will streamline moderation efforts and improve user experience, contributing to the platform's overall quality.



Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)