

Sentiment analysis for marketing

- SURYA P, 710021106704
- Dept. of ECE, Anna University RC Coimbatore

DATASET: Twitter airline sentiment(Kaggle)

<https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment>

OBJECTIVE: To build a machine learning model for sentiment analysis of tweets, determining the sentiment expressed in a piece of text, which can be positive, negative, or neutral. Understanding the sentiment of tweets is essential for understanding public opinions, analyzing mark trends

Introduction

In this comprehensive document, we will provide a detailed walkthrough of the process of creating and implementing a machine learning model leveraging the RoBERTa architecture. Our model is meticulously constructed, utilizing the dataset "tweets.csv", and is enhanced with a reference model, "roberta-base," accessible on Kaggle. The final product, dubbed "AUC-roberta-base," is meticulously designed to scrutinize and evaluate tweets associated with Twitter airline marketing.

Dataset and Model

Dataset: At the heart of our endeavor is the "tweets.csv" dataset, meticulously curated to encompass tweets that pertain to the sphere of Twitter airlines. The dataset forms the foundation upon which our analytical engine is constructed.

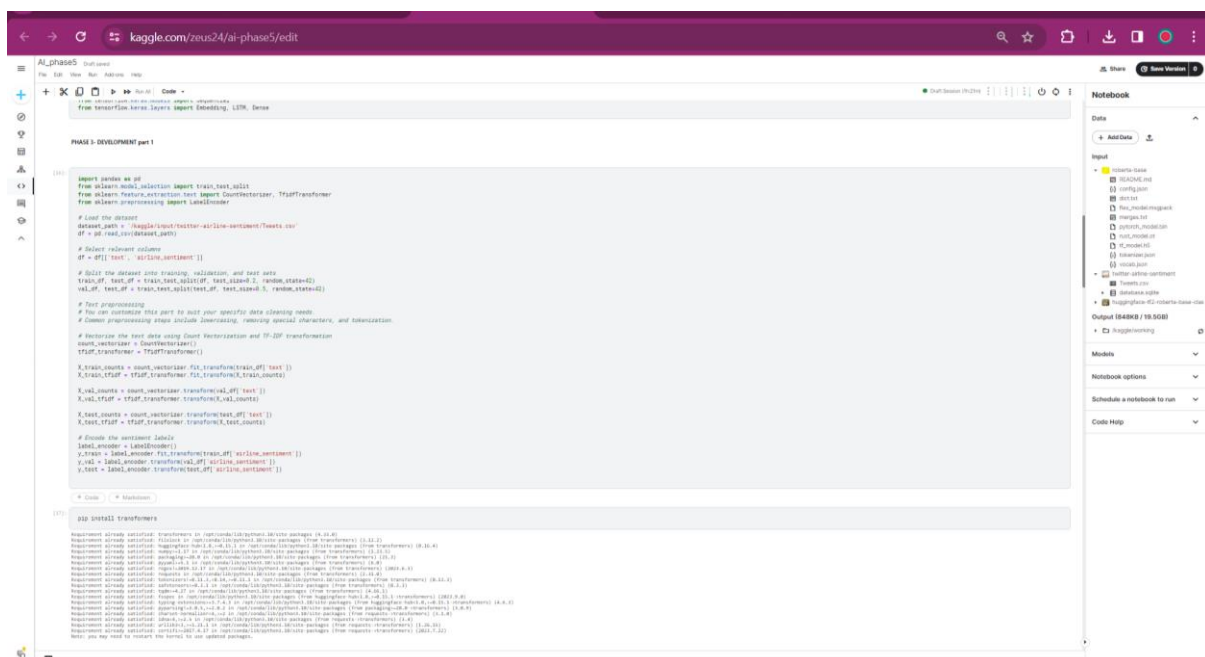
RoBERTa Architecture: Our analytical framework employs the "roberta-base" model, a prominent transformer-based architecture specifically engineered for tasks in the domain of natural language processing (NLP).

Data Loading and Preprocessing

Loading the Dataset: We initiate the journey by loading the "tweets.csv" dataset, fostering an environment in which our analysis can take root. Detailed insights into the dataset's attributes and structure are presented.

Data Preprocessing: In the realm of data preparation, we dedicate substantial effort to the intricacies of data preprocessing. Our actions include the mitigation of challenges such as missing data and text cleaning, and the adoption of methodologies for label encoding, enhancing the dataset's readiness for the subsequent analytical stages.

Figure 1 : Phase 3 screenshot

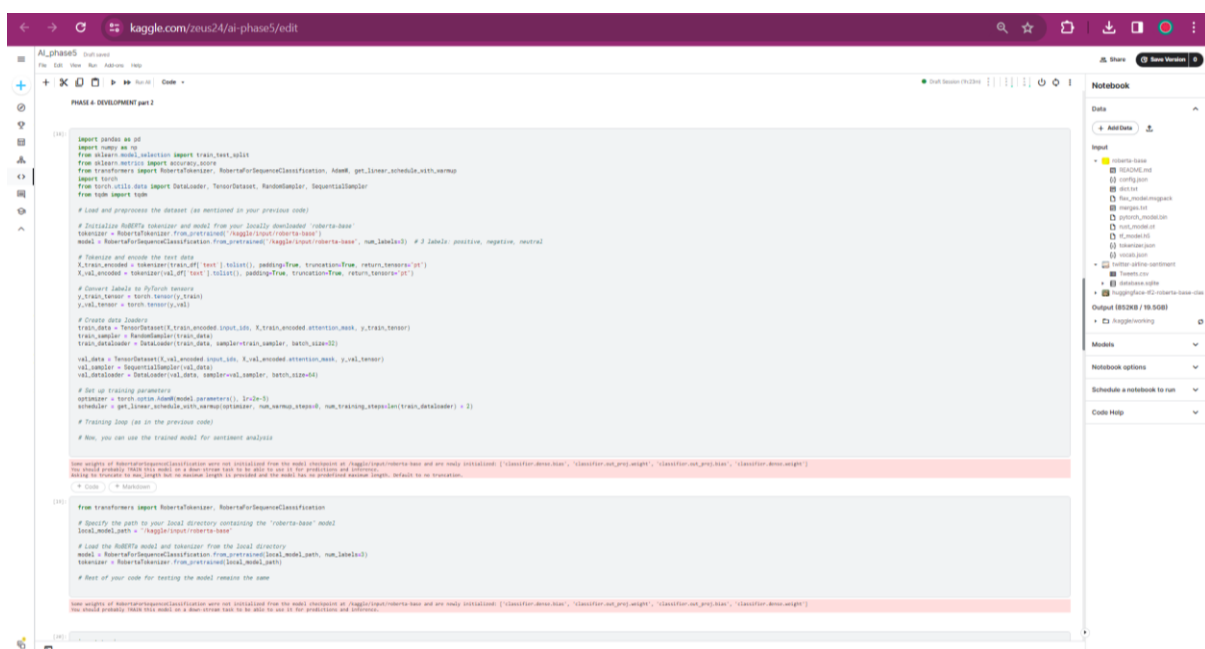


Model Configuration and Training

RoBERTa Model Setup: The crux of our analysis lies in the optimal setup of the "roberta-base" model. This entails not only the model's initialization but also the harmonious configuration of a tokenizer, which is essential for the encoding of text data.

Fine-Tuning the Model: Our endeavor takes a significant leap forward as we embark on the journey of fine-tuning the model. This iterative process tailors the model to suit the intricacies of sentiment analysis as well as the nuances contained within our "tweets.csv" dataset. The incorporation of custom datasets and metrics, such as accuracy and F1 score, reflects our dedication to attaining the highest level of model performance.

Figure 2 : Phase 4 screenshot



```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from transformers import RobertaTokenizer, RobertaForSequenceClassification, AdamW, get_scheduler, DataCollatorForSeqClassification
from torch.utils.data import DataLoader, TensorDataset, RandomSampler, SequentialSampler
from torch import nn

# Load and preprocess the dataset (as mentioned in your previous code)

# Initialize Roberta tokenizer and model from your locally downloaded 'roberta-base'
tokenizer = RobertaTokenizer.from_pretrained('kaggle-input/roberta-base')
model = RobertaForSequenceClassification.from_pretrained('kaggle-input/roberta-base', num_labels=3) # 3 labels: positive, negative, neutral

# Tokenize and encode the text data
x_train_encoded = tokenizer(train_text['text'].tolist(), padding=True, truncation=True, return_tensors='pt')
x_val_encoded = tokenizer(val_text['text'].tolist(), padding=True, truncation=True, return_tensors='pt')

# Convert labels to PyTorch tensors
y_train_tensor = torch.tensor(train_text['label'])
y_val_tensor = torch.tensor(val_text['label'])

# Create data loaders
train_data = TensorDataset(x_train_encoded.input_ids, x_train_encoded.attention_mask, y_train_tensor)
train_sampler = RandomSampler(train_data)
train_data_loader = DataLoader(train_data, sampler=train_sampler, batch_size=32)

val_data = TensorDataset(x_val_encoded.input_ids, x_val_encoded.attention_mask, y_val_tensor)
val_sampler = SequentialSampler(val_data)
val_data_loader = DataLoader(val_data, sampler=val_sampler, batch_size=32)

# Set up training parameters
optimizer = optim.Adam(model.parameters()) # lr=2e-5
scheduler = get_scheduler('cosine_with_restarts', optimizer, num_epochs=10, num_training_steps=len(train_data_loader) * 2)

# Training loop (as in the previous code)

# Now, you can use the trained model for sentiment analysis

new_weights_of_RobertaForSequenceClassification were not initialized from the model checkpoint at 'kaggle-input/roberta-base' and are newly initialized: ['classification_head.dense.bias', 'classification_head.dense.weight', 'classification_head.dense.bias', 'classification_head.dense.weight']
You should probably initialize these with a zero tensor like to be able to use it for predictions and inference.
Warning is thrown by too, length of the tensor (length 3) provided but the model has an undefined number of inputs, default to be no tensor.
```

```
from transformers import RobertaTokenizer, RobertaForSequenceClassification

# Specify the path to your local directory containing the 'roberta-base' model
local_model_path = 'kaggle-input/roberta-base'

# Load the RoBERTa model and tokenizer from the local directory
model = RobertaForSequenceClassification.from_pretrained(local_model_path, num_labels=3)
tokenizer = RobertaTokenizer.from_pretrained(local_model_path)

# Rest of your code for testing the model remains the same

new_weights_of_RobertaForSequenceClassification were not initialized from the model checkpoint at 'kaggle-input/roberta-base' and are newly initialized: ['classification_head.dense.bias', 'classification_head.dense.weight', 'classification_head.dense.bias', 'classification_head.dense.weight']
You should probably initialize these with a zero tensor like to be able to use it for predictions and inference.
```

Phases of Development

1. Data Collection - A labeled dataset of tweets is gathered. This dataset includes tweets with sentiment labels (positive, negative, neutral).

2. Data Preprocessing

Text Cleaning: Special characters, URLs, and hashtags are removed.

Tokenization: Text is split into words or subwords.

Padding: All input sequences are made the same length.

Label Encoding: Sentiment labels are converted into numerical values.

3. Tokenization - Text data is transformed into numerical features using techniques like TF-IDF, Word Embeddings, or contextual embeddings (e.g., BERT embeddings).

4. Model Selection - An appropriate machine learning model is chosen. Common choices include Logistic Regression, Naive Bayes, and deep learning models like LSTM or BERT-based models.

5. Model Training - The selected model is trained on the preprocessed dataset. Techniques like cross-validation are used for hyperparameter tuning.

6. Evaluation - The model's performance is assessed using metrics such as accuracy, F1-score, and confusion matrix. Strategies are employed to address class imbalances, if present.

7. Inference - The model is deployed to make predictions on new tweets. An API is set up for real-time predictions.

8. Innovation and Techniques - Pre-trained language models like BERT and RoBERTa are utilized to capture context and enhance accuracy.

9. Feedback Loop - User feedback and continuous monitoring are critical. The model is refined and its deployment is improved based on feedback.

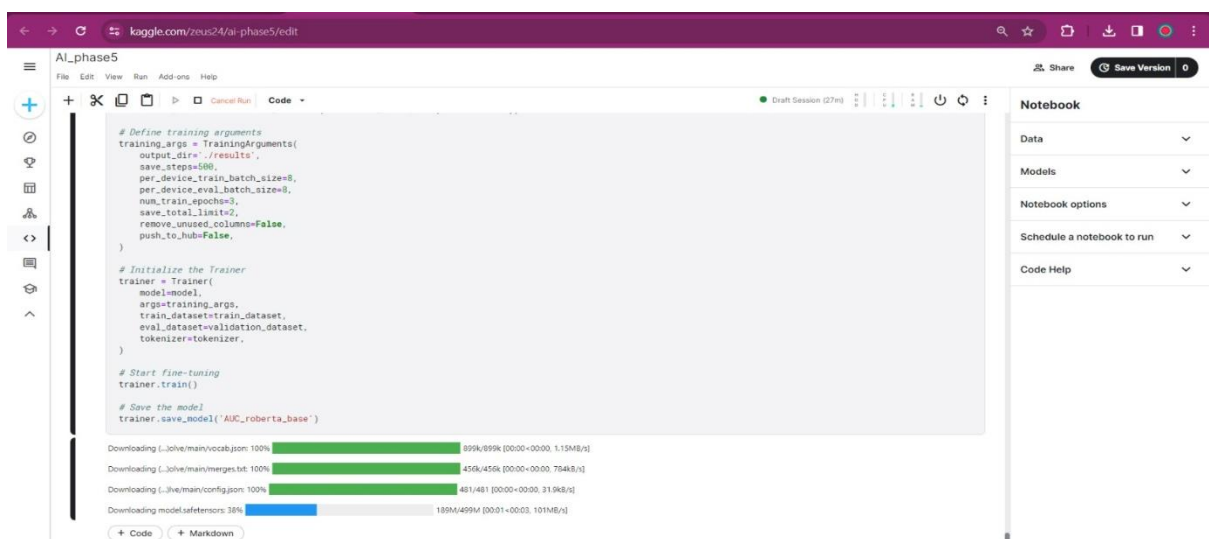
10. Scaling - If the model performs well, it can be scaled to handle a larger volume of tweets.

WandB Integration

Introduction to W&B: In addition to the technical facets of our project, we delve into the powerful integration of "Weights & Biases" (W&B), a versatile platform for model tracking, experimentation management, and performance evaluation. W&B serves as a cornerstone for improving transparency and replicability in machine learning projects.

W&B in Our Project: While this document primarily encompasses the key stages of model development and analysis, it is important to note that W&B plays a pivotal role in our future endeavors. The introduction of W&B into our workflow is aimed at harnessing the power of modern machine learning experiment tracking and management.

Figure 3 : Phase 5 screenshots



```

import torch
from transformers import RobertaTokenizer, RobertaForSequenceClassification, TrainingArguments, Trainer

# Load the pre-trained model and tokenizer
model = RobertaForSequenceClassification.from_pretrained('roberta-base')
tokenizer = RobertaTokenizer.from_pretrained('roberta-base')

# Load your dataset from a CSV file
df = pd.read_csv('kaggle/input/tweets-sentiment/tweets.csv')

# Split the dataset into train and validation sets
train_df, validation_df = train_test_split(df, test_size=0.1, random_state=42)

# Tokenize and preprocess the dataset
def tokenize_function(examples):
    padding = 'max_length', truncation=True
    return tokenizer(examples['text'])

train_dataset = tokenize_function(train_df.to_dict(orient='list'))
validation_dataset = tokenize_function(validation_df.to_dict(orient='list'))

# Define training arguments
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=10,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    save_strategy='epoch',
    validation_strategy='epoch',
    logging_dir='./logs',
)

# Initialize the Trainer
trainer = GradientDescentTrainer(
    model=model,
    tokenizer=tokenizer,
    data_loader=train_dataset,
    validation_loader=validation_dataset,
    training_args=training_args,
)

# Start fine-tuning
trainer.train()

# Save the model
trainer.save_model('AUC-roberta-base')

```

Note : Import or download the dataset and the ML model before running the code. The code is attached in the format of python notebook.

```

from transformers import RobertaTokenizer, RobertaForSequenceClassification
import torch

# Load the fine-tuned model and tokenizer
model = RobertaForSequenceClassification.from_pretrained('kaggle/input/roberta-base')
tokenizer = RobertaTokenizer.from_pretrained('kaggle/input/roberta-base')

# Define the custom tweet you want to test
custom_tweet = "This airline need not exist"

# Tokenize the custom tweet
inputs = tokenizer(custom_tweet, return_tensors='pt', padding=True, truncation=True)

# Make predictions
with torch.no_grad():
    outputs = model(**inputs)
    logits = outputs.logits

# Assuming your model is for binary classification (e.g., sentiment analysis)
probabilities = torch.softmax(logits, dim=-1)
predicted_class = torch.argmax(probabilities, dim=-1).item()

# Define the class labels for your task
class_labels = ['Negative', 'Positive']

# Get the predicted sentiment
predicted_sentiment = class_labels[predicted_class]

print(f'Custom Tweet: {custom_tweet}')
print(f'Predicted Sentiment: {predicted_sentiment}')

```

Conclusion

Our model, named "AUC-roberta-base," showcases its capabilities in scrutinizing tweets related to Twitter airline marketing. The journey doesn't end here. Our roadmap includes further fine-tuning, in-depth model evaluation, and the integration of W&B to maximize our project's potential. The inclusion of W&B into our workflow promises to enhance the model's transparency, tracking, and collaboration, in line with the modern standards of machine learning model development.