SENTIMENT ANALYSIS FOR MARKETING

TEAM MEMBER

821721243042: NITHISH D

821721243023: HARISH S

821721243028: NAVEEN M

821721243007: AJITHKUMAR M

821721243301: AJITH G

Phase 2 Submission Document

Project: Sentiment analysis for marketing (BERT, RoBERTa)

Introduction:

- Sentiment analysis is a Natural language processing technique that identifies the polarity of a given text. There are different flavors of sentiment analysis, but one of the most widely used techniques labels data into positive, negative and neutral.

- Sentiment analysis is a marketing tool that helps you examine the way people interact with a brand online. This method is more comprehensive than traditional online marketing tracking, which measures the number of online interactions that customer.
- Emphasize the need for advanced techniques like fine-tuning pre-trained sentiment analysis models (BERT, RoBERTa) to enhance more accurate sentiment predictions.

Content for Project Phase 2:

- Explore advanced techniques like fine-tuning pre-trained sentiment analysis models (BERT, RoBERTa) for more accurate sentiment predictions.

Data Source:

- A good data source for Sentiment analysis for marketing should be Accurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link:(https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment)

Airline sentiment	Airline sentiment confidence	Negative reason	Negative reason confidence	Airline
neutral	1			Virgin America
positive	0.3486		0	Virgin America
neutral	0.6837			Virgin America
negative	1		0.7033	Virgin America
negative	1	Bad flight	1	Virgin America
negative	1	Can't tell	0.6842	Virgin America
positive	0.6745	Can't tell	0	Virgin America
neutral	0.634			Virgin America
positive	0.6559			Virgin America
positive	1			Virgin America
neutral	0.6769		0	Virgin America
positive	1			Virgin America
positive	1			Virgin America

Data Preparation:

- Collect and preprocess your sentiment analysis dataset. Ensure it is well-labeled with sentiment labels (e.g., positive, negative, neutral).
- Split your dataset into training, validation, and test sets.

Select a Pre-trained Model:

- Choose a pre-trained model that suits your needs. BERT, RoBERTa, GPT-2, and others are popular choices.
- You can use pre-trained models available through Hugging Face Transformers library, which provides pre-trained models and easy-to-use APIs for fine-tuning.

Model Architecture:

- Modify the architecture of the pre-trained model to suit the specific sentiment analysis task.
- Add a classification layer on top of the model. The number of output units should match the number of sentiment classes in your dataset.

Fine-Tuning:

- Fine-tuning involves training the selected pre-trained model on your sentiment analysis dataset.
- Use transfer learning, where you start with the weights learned during pre-training and fine-tune them on your task-specific data.
- Train the model on your training data and validate it on the validation set.

Hyperparameter Tuning:

- Experiment with different hyperparameters such as learning rates, batch sizes, and the number of epochs to optimize model performance.
- Utilize techniques like learning rate scheduling to improve training stability.

Loss Function:

- Choose an appropriate loss function for your sentiment classification task. Cross-entropy loss is commonly used.

Regularization Techniques:

- Employ techniques like dropout and weight decay to prevent overfitting.

Evaluation:

- Assess the fine-tuned model's performance on the test dataset using metrics like accuracy, precision, recall, F1-score, and ROC AUC.

Post-processing:

- Depending on your specific application, you may want to perform post-processing on model predictions, such as thresholding or applying smoothing techniques.

Model Deployment:

- Once you have a well-fine-tuned sentiment analysis model, deploy it in your desired application, whether it's for real-time sentiment analysis on social media or integrating it into a customer support chat bot.

Model Monitoring and Maintenance:

- Continuously monitor the model's performance in production to ensure it remains accurate over time.
- Periodically re-train the model with new data to adapt to evolving language and sentiment trends.

Ethical Considerations:

- Be mindful of potential biases in the data and the model. Implement fairness and bias mitigation techniques, and ensure the model aligns with ethical guidelines.

Program:

Sentiment analysis for marketing

```
from textblob import TextBlob

def analyze_sentiment(text):
analysis = TextBlob(text)
if analysis.sentiment.polarity &gt; 0:
return &#39;Positive&#39;
elif analysis.sentiment.polarity == 0:
return &#39;Neutral&#39;
else:
return &#39;Negative&#39;

# Example usage
text = &quot;Stock prices are rising, and investors are optimistic about the
market.&quot;
sentiment = analyze_sentiment(text)
print(&quot;Sentiment:&quot;, sentiment)
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
```

```
# Load your labeled dataset
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, labels,
test_size=0.2, random_state=42)

# Vectorize the text data
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

# Train a Naive Bayes classifier
clf = MultinomialNB()
clf.fit(X_train_vectorized, y_train)

# Make predictions
predictions = clf.predict(X_test_vectorized)

# Calculate accuracy
accuracy = accuracy_score(y_test, predictions)
print(&quot;Accuracy:&quot;, accuracy)
```

Model 1 - Load the Pre-trained Language Model and Tokenizer:

```
from transformers import DistilBertTokenizer,
DistilBertForSequenceClassification


# Load the pre-trained tokenizer
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')


# Load the pre-trained model for sequence classification
model = DistilBertForSequenceClassification.from_pretrained('distilbert-
base-uncased')
```

Model 2 - Prepare the Sentiment Analysis Dataset:

```
texts = ["I loved the movie. It was great!",
        "The food was terrible.",
        "The weather is okay."]
sentiments = ["positive", "negative", "neutral"]

# Tokenize the text samples
encoded_texts = tokenizer(texts, padding=True, truncation=True,
return_tensors='pt')


# Extract the input IDs and attention masks
input_ids = encoded_texts['input_ids']
attention_mask = encoded_texts['attention_mask']
```

```
# Convert the sentiment labels to numerical form
sentiment_labels = [sentiments.index(sentiment) for sentiment in sentiments]
```

Model 3 - Add a Custom Classification Head:

```
import torch.nn as nn

# Add a custom classification head on top of the pre-trained model
num_classes = len(set(sentiment_labels))
classification_head = nn.Linear(model.config.hidden_size, num_classes)

# Replace the pre-trained model's classification head with our custom head
model.classifier = classification_head
```

Model 4 - Fine-Tune the Model:

```
import torch.optim as optim

# Define the optimizer and loss function
optimizer = optim.AdamW(model.parameters(), lr=2e-5)
criterion = nn.CrossEntropyLoss()

# Fine-tune the model
num_epochs = 3
for epoch in range(num_epochs):
```

```
    optimizer.zero_grad()
    outputs = model(input_ids, attention_mask=attention_mask,
labels=torch.tensor(sentiment_labels))
    loss = outputs.loss
    loss.backward()
    optimizer.step()
```

Conclusion

- In the Phase 2 conclusion, we will summarize the key findings and insights from the advanced techniques. We will reiterate the impact of these techniques on improving the accuracy and robustness of Sentiment analysis for marketing.

*****THANK YOU*****