SENTIMENT ANALYSIS FOR MARKETING

Problem Definition :

The problem at hand is to perform sentiment analysis on customer feedback to gain insights into competitor products. Understanding customer sentiments is crucial for companies to identify strengths and weaknesses in competing products, thereby enhancing their offerings. This project involves the utilization of various Natural Language Processing (NLP) methods to extract valuable insights from customer feedback.

Data Collection :

Start by gathering customer feedback data related to competitor products. This data can be collected from various sources, such as online reviews, surveys, social media, or customer support interactions.

Data Preprocessing :

Clean and preprocess the collected data. This involves tasks like text normalization, removing stopwords, and handling special characters or emojis.

Text Tokenization :

Tokenize the text data into words or phrases, which is a fundamental step in NLP.

Sentiment Analysis Model :

You can use pre-trained models like BERT, GPT-3, or train your own sentiment analysis model using machine learning techniques like LSTM or CNN.

Labeling Data :

If you don't already have labeled data indicating the sentiment (positive, negative, or neutral), you may need to manually label a portion of the data for training and validation.

Model Training :

Train your sentiment analysis model on the labeled data. Fine-tune pre-trained models if needed.

Evaluation :

Assess the performance of your model using metrics like accuracy, precision, recall, and F1-score.

Generating Insights :

Once your model is ready, you can analyze the sentiment of customer feedback. Insights could include identifying common positive and negative aspects of competitor products, tracking sentiment trends over time, or comparing products' sentiment.

Visualization :

Visualize your insights using charts or graphs to make them more understandable for stakeholders.

Iterate :

Continue to improve your model and insights by iterating on the process, incorporating user feedback, and adapting to changing customer sentiments.

Dataset Handling :

The code begins by loading a dataset from a CSV file using Pandas and splitting it into training and testing sets. In this case, the dataset is the Twitter US Airline Sentiment dataset, containing tweets and corresponding sentiment labels (positive, negative, or neutral).

BERT Model and Tokenizer :

It initializes a BERT model and tokenizer using the Hugging Face Transformers library. BERT is a powerful pre-trained model that can understand contextual information in text.

Training BERT Model :

The code sets up training parameters and fine-tunes the BERT model for the sentiment classification task. This involves specifying batch sizes, the number of training epochs, and evaluation intervals.

Prediction :

After fine-tuning, the code uses a BERT-based pipeline for sentiment analysis to make predictions on the test dataset.

Evaluation :

The code uses standard classification metrics like accuracy and the classification report (including precision, recall, and F1-score) to evaluate the model's performance on sentiment classification.

Fine-Tuning for Specific Tasks :

Fine-tuning is crucial when using pre-trained models like BERT. It adapts the model to your specific task, in this case, sentiment analysis on airline tweets.

Adjusting for Your Dataset :

Using BERT represents a more advanced NLP technique, as it can capture complex language patterns and context. It tends to perform well on various NLP tasks, including sentiment analysis.

PROGRAM :

sentiment analysis project with NLP techniques

Install Libraries :

```
pip install pandas scikit-learn transformers
```

Code for Sentiment Analysis with BERT :

```
import pandas as pd
from sklearn.model_selection import train_test_split
from transformers import BertTokenizer,
BertForSequenceClassification, pipeline

# Load the dataset
df = pd.read_csv('Tweets.csv')

# Split the data into training and testing sets
X = df['text']
y = df['airline_sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Initialize BERT tokenizer and model
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-
uncased', num_labels=3)

# Encode text data
X_train_encodings = tokenizer(list(X_train), padding=True,
truncation=True, return_tensors='pt', max_length=128)
```

```
X_test_encodings = tokenizer(list(X_test), padding=True,
truncation=True, return_tensors='pt', max_length=128)

# Train a BERT-based classifier
from transformers import Trainer, TrainingArguments

training_args = TrainingArguments(
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    output_dir='./results',
    evaluation_strategy="steps",
    num_train_epochs=3,
    save_steps=10,
    eval_steps=10,
)

trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=None,
    train_dataset=None, # Pass your training dataset here
    eval_dataset=None, # Pass your evaluation dataset here
)
```

```
# Make predictions on the test set
sentiment_pipeline = pipeline(task="sentiment-analysis", model=model, tokenizer=tokenizer)
y_pred = sentiment_pipeline(list(X_test))

# Evaluate the model
from sklearn.metrics import accuracy_score, classification_report

y_pred_labels = [item['label'] for item in y_pred]
accuracy = accuracy_score(y_test, y_pred_labels)
print(f"Accuracy: {accuracy:.2f}")

print(classification_report(y_test, y_pred_labels))
```

conclusion :

This project aims to leverage NLP techniques to analyze customer feedback on competitor products, helping companies make data-driven decisions. The outlined design thinking process encompasses data collection, preprocessing, sentiment analysis, feature extraction, visualization, and insights generation.