NLP-Sentiment analysis for marketing

MODEL TRAINING:

- In marketing, sentiment analysis involves training machine learning models on vast datasets of customer feedback and social media content to automatically assess and categorize sentiments as positive, negative, or neutral.
- This training enables marketers to gain valuable insights into consumer opinions and emotions, helping tailor campaigns and strategies to better align with customer sentiment.

```
import pandas as pd
import numpy as np
import os
import random
from pathlib import Path
import json


import torch
from tqdm.notebook import tqdm
from transformers import BertTokenizer
from torch.utils.data import TensorDataset
from transformers import BertForSequenceClassification

class Config():
    seed_val = 17
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    epochs = 5
    batch_size = 6
    seq_length = 512
    lr = 2e-5
    eps = 1e-8
```

```
    pretrained_model = 'bert-base-uncased'
    test_size=0.15
    random_state=42
    add_special_tokens=True
    return_attention_mask=True
    pad_to_max_length=True
    do_lower_case=False
    return_tensors='pt'

config = Config()
params = {"seed_val": config.seed_val,
    "device":str(config.device),
    "epochs":config.epochs,
    "batch_size":config.batch_size,
    "seq_length":config.seq_length,
    "lr":config.lr,
    "eps":config.eps,
    "pretrained_model": config.pretrained_model,
    "test_size":config.test_size,
    "random_state":config.random_state,
    "add_special_tokens":config.add_special_tokens,
    "return_attention_mask":config.return_attention_mask,
    "pad_to_max_length":config.pad_to_max_length,
    "do_lower_case":config.do_lower_case,
    "return_tensors":config.return_tensors,
        }
linkcode

import random
device = config.device
random.seed(config.seed_val)
np.random.seed(config.seed_val)
```

```
torch.manual_seed(config.seed_val)
torch.cuda.manual_seed_all(config.seed_val)
df.head()
```

Train and Validation Split

```
from sklearn.model_selection import train_test_split
train_df_, val_df = train_test_split(df,  test_size=0.10,
    random_state=config.random_state, stratify=df.label.values)
```

linkcode

```
train_df_.head()
```

Creating the Model

- bert-base-uncased is a smaller pre-trained model.
- Using num_labels to indicate the number of output labels.

```
model =    BertForSequenceClassification.from_pretrained(config.pretrained_model,
num_labels=3,
output_attentions=False,                                    output_hidden_states=False)
```

Data Loaders

```
from torch.utils.data import DataLoader, RandomSampler, SequentialSampler
dataloader_train = DataLoader(dataset_train,
                              sampler=RandomSampler(dataset_train),
                              batch_size=config.batch_size)
dataloader_validation = DataLoader(dataset_val,
sampler=SequentialSampler(dataset_val),
```

```
                                batch_size=config.batch_size)
```

Optimizer & Scheduler

```
from transformers import AdamW, get_linear_schedule_with_warmup
optimizer = AdamW(model.parameters(),
                  lr=config.lr,
                  eps=config.eps)
scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps=0,
num_training_steps=len(dataloader_train)*config.epochs)
```

ERROE ANALYSIS

- Error analysis is a critical step in machine learning model development where researchers and data scientists meticulously investigate and categorize prediction errors to identify patterns, improve model performance, and refine training data or features for enhanced accuracy.
- It plays a crucial role in fine-tuning models and optimizing their real-world applicability.

```
pred_final = []

for i, row in tqdm(val_df.iterrows(), total=val_df.shape[0]):
    predictions = []
    review = row["Review"]
    encoded_data_test_single = tokenizer.batch_encode_plus(
    [review],
    add_special_tokens=config.add_special_tokens,
    return_attention_mask=config.return_attention_mask,
    pad_to_max_length=config.pad_to_max_length,
    max_length=config.seq_length,
    return_tensors=config.return_tensors
    )
    input_ids_test = encoded_data_test_single['input_ids']
    attention_masks_test = encoded_data_test_single['attention_mask']


    inputs = {'input_ids':      input_ids_test.to(device),
              'attention_mask':attention_masks_test.to(device),
             }

    with torch.no_grad():
        outputs = model(**inputs)

    logits = outputs[0]
    logits = logits.detach().cpu().numpy()
    predictions.append(logits)
    predictions = np.concatenate(predictions, axis=0)
    pred_final.append(np.argmax(predictions, axis=1).flatten()[0])
```

INFERENCE

- In sentiment analysis, inference refers to the process of using a trained model to make predictions about the sentiment of text or content, such as determining whether a customer review is positive or negative.
- This inference step is crucial for automating sentiment classification tasks and extracting actionable insights from textual data in marketing, customer service, and other domains.

```
test_df.head()
encoded_data_test = tokenizer.batch_encode_plus(
    test_df.Review.values,
    add_special_tokens=config.add_special_tokens,
    return_attention_mask=config.return_attention_mask,
    pad_to_max_length=config.pad_to_max_length,
    max_length=config.seq_length,
    return_tensors=config.return_tensors
)
input_ids_test = encoded_data_test['input_ids']
attention_masks_test = encoded_data_test['attention_mask']
labels_test = torch.tensor(test_df.label.values)
```

linkcode

```
model = BertForSequenceClassification.from_pretrained(config.pretrained_model,
                                      num_labels=3,
                                      output_attentions=False,
                                      output_hidden_states=False)

model.to(config.device)

model.load_state_dict(torch.load(f'./_BERT_epoch_3.model', map_location=torch.device('cpu')))

_, predictions_test, true_vals_test = evaluate(dataloader_validation)
```

Conclusion

- Sentiment analysis in marketing is a powerful tool that helps businesses make data-driven decisions, enhancing customer engagement and product positioning based on a deeper understanding of consumer sentiments, ultimately leading to more effective marketing strategies.