Aim: To build an adaptive and contextual cognitive-based application for customer service, insurance, healthcare, smarter cities, or government.

Theory:

What is an Adaptive and Contextual Cognitive Application?

An adaptive application is one that can adjust its behavior based on the user's past interactions or current input. A contextual application understands the situation or mood of the user and gives responses accordingly.

In this experiment, we have created a smart assistant that understands how the user feels about their electricity usage (like high bills or complaints) and gives meaningful suggestions based on their sentiment.

This kind of system is very useful in smarter cities where users expect personalized advice and intelligent customer service.

What is Sentiment Analysis?

Sentiment analysis is a technique used to find out whether the user's feedback is positive, negative, or neutral. It helps the system understand the emotion behind the message.

We used three different techniques to do this:

1. TextBlob – A simple tool to calculate polarity (how positive or negative a sentence is) and subjectivity (whether it is opinion or fact).
2. VADER – A tool specially made for short messages, social media text, and customer feedback.
3. Transformer-based model – An advanced AI model that gives more accurate sentiment detection using deep learning.

How the Application Works:

1. The user types something related to electricity usage (e.g., “My bill is too high this month”).
2. The system uses TextBlob, VADER, and Transformers to understand if the message is:
    - Positive (e.g., “I am happy with my bill”)
    - Negative (e.g., “This bill is too much”)
    - Neutral (e.g., “Please show my usage”)
3. Based on the result, the system gives a custom response:
    - Positive → Encouragement and tips to maintain savings
    - Negative → Apology and suggestions to reduce energy usage
    - Neutral → Generic reply or more information

Why It Is Adaptive and Contextual

- The application is adaptive because it changes its response based on how the user feels.

- It is contextual because it understands the user's situation (like frustration with high bills or satisfaction with savings) and replies meaningfully.
- The same question can get different answers depending on how the user says it.

Real-World Use Case: Smart Electricity in Smarter Cities

In a smart city:

- Customers can chat with such a system to get suggestions to save energy
- The system can learn from user feedback and adjust its behavior
- Utility companies can improve service without manual support

```
# Install dependencies
!pip install textblob vaderSentiment transformers --quiet
```

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
from textblob import TextBlob
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from transformers import pipeline
```

```
# Initialize sentiment analyzers
vader_analyzer = SentimentIntensityAnalyzer()
transformer_pipeline = pipeline("sentiment-analysis")
```

```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english and revision
714eb0f (https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is not recommended.
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(

 config.json:  100%

  629/629  [00:00<00:00,  14.7kB/s]

 model.safetensors:  100%

  268M/268M  [00:02<00:00,  122MB/s]

 tokenizer_config.json:  100%

  48.0/48.0  [00:00<00:00,  2.73kB/s]

 vocab.txt:

  232k/?  [00:00<00:00,  11.8MB/s]

Device set to use cpu
```

```
# Context-aware responses
def get_adaptive_response(sentiment, polarity=None):
    if sentiment == "POSITIVE" or polarity and polarity > 0.3:
        return "Glad to hear that! Keep using energy-efficient appliances
to maintain low bills."
    elif sentiment == "NEGATIVE" or polarity and polarity < -0.3:
```

```
        return "Sorry to hear that. Try turning off unused devices and consider switching to solar solutions."
    else:
        return "Thanks for your input. Let me know if you want tips to reduce electricity usage."
```

```
# Input feedback or queries
user_input = input("You: ")
```

```
You: My electricity bill is too high this month.
```

```
# 1. TextBlob Analysis
blob = TextBlob(user_input)
polarity = blob.sentiment.polarity
subjectivity = blob.sentiment.subjectivity
```

```
print("\nTextBlob Analysis:")
print("Polarity:", polarity)
print("Subjectivity:", subjectivity)
print("→", get_adaptive_response(None, polarity))
```

```
TextBlob Analysis:
Polarity: 0.16
Subjectivity: 0.54
→ Thanks for your input. Let me know if you want tips to reduce electricity usage.
```

```
#2. VADER Analysis
vader_scores = vader_analyzer.polarity_scores(user_input)
print("\nVADER Analysis:")
print("Scores:", vader_scores)
compound = vader_scores['compound']
if compound >= 0.05:
    print("Sentiment: POSITIVE")
elif compound <= -0.05:
    print("Sentiment: NEGATIVE")
else:
    print("Sentiment: NEUTRAL")
```

```
VADER Analysis:
Scores: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Sentiment: NEUTRAL
```

```
# 3. Transformers Analysis
transformer_result = transformer_pipeline(user_input)[0]
print("\nTransformer Model Analysis:")
print("Label:", transformer_result['label'])
print("Score:", transformer_result['score'])
print("→", get_adaptive_response(transformer_result['label']))
```

```
Transformer Model Analysis:
Label: NEGATIVE
Score: 0.9995452761650085
→ Sorry to hear that. Try turning off unused devices and consider switching to solar
solutions.
```

Conclusion:

This experiment shows how a cognitive application can become smart, adaptive, and emotionally aware by using sentiment analysis. It helps provide better customer service and energy-saving advice by understanding the mood and intent of the user. This approach can be used in customer care, smart city utilities, and many other areas where personalized and intelligent interaction is required.

AI&DS2_Expt_04