2. NLTK-Powered Text Analytics Web App (Flask/Streamlit + pandas)

(i)nlp_pipeline.py

```
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag, FreqDist
from nltk.corpus import stopwords
from nltk.collocations import BigramCollocationFinder, BigramAssocMeasures
from textblob import TextBlob
import string

nltk.download("punkt")
nltk.download("stopwords")
nltk.download("averaged_perceptron_tagger")

def clean_text(text):
    tokens = word_tokenize(text.lower())
    stop_words = set(stopwords.words("english"))
    tokens = [word for word in tokens if word not in stop_words and word.isalpha()]
    return tokens

def analyze_text(text):
    tokens = clean_text(text)
    tagged = pos_tag(tokens)
    fdist = FreqDist(tokens)

    bigram_finder = BigramCollocationFinder.from_words(tokens)
    bigrams = bigram_finder.nbest(BigramAssocMeasures.likelihood_ratio, 5)

    sentiment = TextBlob(text).sentiment

    return {
        "tokens": tokens,
        "tagged": tagged,
        "fdist": fdist,
        "bigrams": bigrams,
        "sentiment": sentiment
    }
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

(ii)streamlit_app.py

```
# Create the nlp_pipeline.py file
with open("nlp_pipeline.py", "w") as f:
    f.write("""
```

```
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag, FreqDist
from nltk.corpus import stopwords
from nltk.collocations import BigramCollocationFinder, BigramAssocMeasures
from textblob import TextBlob
import string

nltk.download("punkt")
nltk.download("stopwords")
nltk.download("averaged_perceptron_tagger")

def clean_text(text):
    tokens = word_tokenize(text.lower())
    stop_words = set(stopwords.words("english"))
    tokens = [word for word in tokens if word not in stop_words and word.isalpha()]
    return tokens

def analyze_text(text):
    tokens = clean_text(text)
    tagged = pos_tag(tokens)
    fdist = FreqDist(tokens)

    bigram_finder = BigramCollocationFinder.from_words(tokens)
    bigrams = bigram_finder.nbest(BigramAssocMeasures.likelihood_ratio, 5)

    sentiment = TextBlob(text).sentiment

    return {
        "tokens": tokens,
        "tagged": tagged,
        "fdist": fdist,
        "bigrams": bigrams,
        "sentiment": sentiment
    }
""")

# Create the streamlit_app.py file
with open("streamlit_app.py", "w") as f:
    f.write("""
import streamlit as st
from nlp_pipeline import analyze_text
import matplotlib.pyplot as plt

st.title(" NLP Text Analyzer (NLTK + Streamlit)")

uploaded_file = st.file_uploader("Upload a .txt file", type=["txt"])

if uploaded_file is not None:
    text = uploaded_file.read().decode("utf-8")
    st.subheader("Uploaded Text")
    st.write(text)

    st.subheader("Analysis Results")
    result = analyze_text(text)

    st.write("**Top 10 Words (by Frequency):**")
    freq_words = result["fdist"].most_common(10)
    for word, count in freq_words:
        st.write(f"{word}: {count}")
```

```
    st.write("**Top 5 Bigrams:**")
    st.write(result["bigrams"])

    st.write("**Sentiment Analysis:**")
    st.write(f"Polarity: {result['sentiment'].polarity}")
    st.write(f"Subjectivity: {result['sentiment'].subjectivity}")

    # Plot frequency
    st.subheader("Word Frequency Chart")
    fig, ax = plt.subplots()
    words, counts = zip(*freq_words)
    ax.bar(words, counts, color="skyblue")
    plt.xticks(rotation=45)
    st.pyplot(fig)
""")
```

(iii)example.txt

```
with open("example.txt", "w") as f:
    f.write("""I love using natural language processing. It helps us analyze text data.
Sometimes the results are positive, and sometimes they are negative.
This tool is amazing for research and fun to use!""")
```

(iv)README.md

Task 2: Text Analytics Web App (NLTK + Streamlit)

Description

A simple web app that performs text analysis:

- Tokenization
- POS Tagging
- Frequency Distribution
- Bigrams
- Sentiment Analysis

Tech Stack

- Streamlit
- NLTK
- TextBlob
- Matplotlib