Intelligence System:

Natural Language Processing (NLP)

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

Intelligence System Development

2024 – 2025
Y4E1 – DCS – NU

By: SEK SOCHEAT
Advisor to DCS and Lecturer
Mobile: 017 879 967
Email: socheat.sek@gmail.com

Table of Contents

1. Introduction to NLP for Intelligence Systems

1.1. Overview of Natural Language Processing

1.2. Importance of NLP

1.3. Applications of NLP

1.5. Key NLP Tasks

Example: Grammar-Based English-Khmer Translation System

2. Integrating NLP with Intelligence Systems

2.1. How NLP Enhances

2.2. Overview of Text-Based

2.3. Architecture of NLP-Integrated

3. Practical: Grammar-Based English-Khmer Translation System

4. Homework

1. Introduction to NLP

- Natural Language Processing (NLP) has become a cornerstone in artificial intelligence for systems emulating human decision-making, including intelligence systems.
- It allows machines to interpret, generate, and interact using human language, creating opportunities for seamless integration into domains like healthcare, finance, and customer support.

1. Introduction to NLP

1.1 Overview of Natural Language Processing

- NLP bridges human communication with machine understanding, processing text and speech into actionable insights.
- It involves parsing sentences, understanding semantics, and generating coherent responses.
- Techniques include syntactic parsing, tokenization, and vector representations of language, supported by tools like spaCy, NLTK, and advanced transformer models such as BERT and GPT.

1. Introduction to NLP

1.2. Importance of NLP

1. Natural Interaction: Enables communication using everyday language, enhancing usability.

2. Knowledge Extraction: Processes unstructured text to extract valuable insights for decision-making.

3. Real-Time Decision Support: Provides instant, accurate responses based on linguistic analysis.

4. Context Understanding: Handles ambiguity and ensures precise interpretation of user queries.

5. Domain Adaptation: Tailors expertise by analyzing specific language and terminology of the field.

6. Dynamic Learning: Updates knowledge bases by parsing new text sources automatically.

NLP empowers intelligence systems to emulate human reasoning and interaction effectively.

1. Introduction to NLP

1.3 Applications of NLP

In intelligence systems, NLP enhances interactions, enabling:

1. Diagnosis Assistance: Systems like MYCIN use linguistic data for medical inference.

2. Legal and Contract Analysis: Automating comprehension and clause extraction in legal documents.

3. Customer Support: Chatbots that provide domain-specific expertise, responding conversationally to queries.

1. Introduction to NLP

1.4. Key NLP Tasks

For integration into intelligence systems, key NLP tasks include:

1. Information Retrieval: Extracting domain-relevant details.

2. Semantic Understanding: Recognizing relationships and context in queries.

3. Dialogue Systems: Building interactive frameworks for query and response handling.

4. Sentiment and Intent Analysis: Capturing user intention for personalized system guidance.

1. Introduction to NLP

Example: Grammar-Based English-Khmer Translation System

Objective:

To create a rule-based translation system that converts English text into Khmer (and vice versa) by:

1. Parsing English grammar structures using predefined linguistic rules.
2. Applying grammar translation rules to produce accurate Khmer equivalents.

Key Components:

1. Grammar Rules for English:

- Define sentence structures (e.g., Subject-Verb-Object patterns).
- Identify grammatical categories (e.g., nouns, verbs, adjectives).

2. Grammar Translation Rules:

- Map English grammar structures to Khmer grammar rules.
- Account for linguistic differences, such as word order (e.g., SVO in English → SOV in Khmer).

3. Core Language (English):

- English serves as the intermediary to ensure universal grammar rules before mapping to Khmer.

1. Introduction to NLP

Example: Grammar-Based English-Khmer Translation System

Example Translation Workflow:

1. Input (English):

- "I love learning."

2. Grammar Parsing:

- ***Tokenization:*** ["I", "love", "learning"]
- ***Grammar Structure:*** Subject → Verb → Object

3. Rule-Based Mapping (English → Khmer):

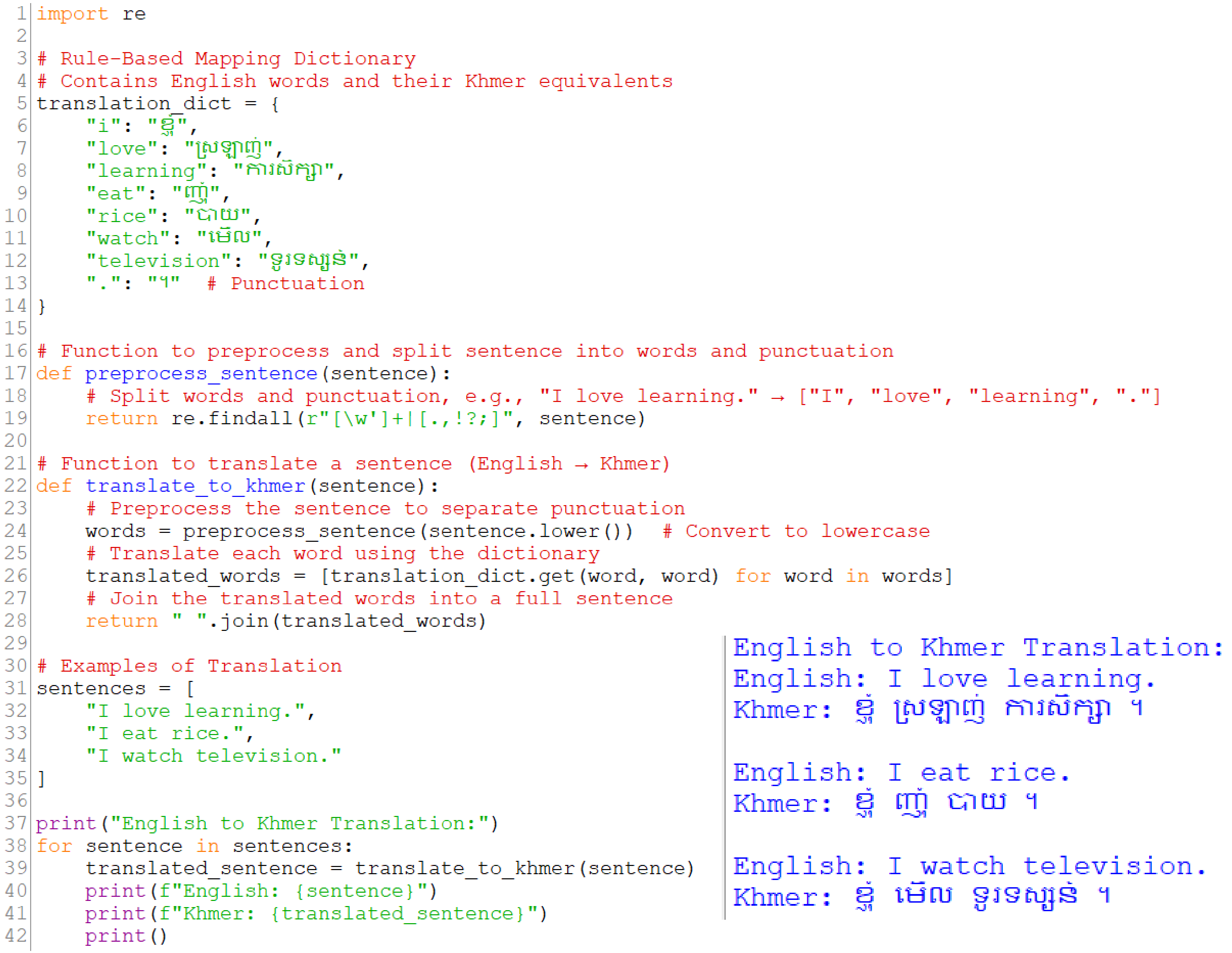
- Subject-Verb-Object in English → Subject-Verb-Object in Khmer.
- "ខ្ញុំ" (I) → "ស្រឡាញ់" (love) → “ការសិក្សា។" (learning.)

4. Output (Khmer):

- "ខ្ញុំ ស្រឡាញ់ ការសិក្សា។"

Example:

Grammar-Based English-Khmer Translation System



```python
import re

# Rule-Based Mapping Dictionary
# Contains English words and their Khmer equivalents
translation_dict = {
    "i": "ខ្ញុំ",
    "love": "ស្រឡាញ់",
    "learning": "ការសិក្សា",
    "eat": "ញ៉ាំ",
    "rice": "បាយ",
    "watch": "មើល",
    "television": "ទូរទស្សន៍",
    ".": "។"  # Punctuation
}

# Function to preprocess and split sentence into words and punctuation
def preprocess_sentence(sentence):
    # Split words and punctuation, e.g., "I love learning." → ["I", "love", "learning", "."]
    return re.findall(r"[\w']+|[.,!?;]", sentence)

# Function to translate a sentence (English → Khmer)
def translate_to_khmer(sentence):
    # Preprocess the sentence to separate punctuation
    words = preprocess_sentence(sentence.lower())  # Convert to lowercase
    # Translate each word using the dictionary
    translated_words = [translation_dict.get(word, word) for word in words]
    # Join the translated words into a full sentence
    return " ".join(translated_words)

# Examples of Translation
sentences = [
    "I love learning.",
    "I eat rice.",
    "I watch television."
]

print("English to Khmer Translation:")
for sentence in sentences:
    translated_sentence = translate_to_khmer(sentence)
    print(f"English: {sentence}")
    print(f"Khmer: {translated_sentence}")
    print()
```

```
English to Khmer Translation:
English: I love learning.
Khmer: ខ្ញុំ ស្រឡាញ់ ការសិក្សា ។

English: I eat rice.
Khmer: ខ្ញុំ ញ៉ាំ បាយ ។

English: I watch television.
Khmer: ខ្ញុំ មើល ទូរទស្សន៍ ។
```

2. Integrating NLP

- Integrating NLP with intelligence systems bridges human communication and machine reasoning, enabling intuitive and interactive problem-solving.
- This synergy enhances system usability, automates text-based data extraction, and provides more accurate decision support.

2. Integrating NLP

2.1. How NLP Enhances

1. **Human-Like Interaction:** Allows users to communicate in natural language, eliminating technical barriers.

2. **Automated Knowledge Extraction:** Processes large volumes of unstructured data to enrich the intelligence system’s knowledge base.

3. **Context-Aware Decision-Making:** Leverages semantic understanding for nuanced recommendations.

2. Integrating NLP

2.2. Overview of Text-Based

1. Definition: Systems that rely on textual inputs (e.g., documents, queries) to provide domain-specific expertise.

2. Workflow: NLP processes input text, extracts relevant features, and integrates insights into the intelligence system’s inference engine.

3. Core Components: Text parser, semantic analyzer, knowledge base, and reasoning module.

2. Integrating NLP

2.3. Architecture of NLP-Integrated

An NLP-integrated itelligence system has two main components:

1. NLP Pipeline:

- Processes user queries by tokenizing, lemmatizing, and extracting keywords or entities.
- Converts unstructured text into structured data for system interpretation.

2. Knowledge Base or Rule System:

- Stores domain-specific facts, rules, or heuristics.
- Uses the structured input from the NLP pipeline to infer answers or provide recommendations via a reasoning engine.

This architecture ensures seamless text understanding and decision-making.

2. Integrating NLP

Example of Building an NLP Pipeline

Building an NLP pipeline involves:

1. Data Preprocessing: Tokenization, lemmatization, and removing noise.

2. Feature Extraction: Representing words or sentences using embeddings like word2vec or transformers.

3. Modeling: Applying neural networks, such as recurrent networks or transformers, to learn domain-specific tasks.

4. Evaluation: Validating performance using metrics like BLEU for generated text or F1 score for classification tasks.

Example: Sentiment Analyzer with Dynamic Dataset Management

Objective:

To design and implement a Sentiment Analysis System that allows users to:

- **Classify Sentiments:** Analyze user-provided sentences to predict their sentiment (e.g., Positive, Neutral, Negative).
- **Manage Dataset Dynamically:**
  - Perform CRUD operations (Create, Read, Update, Delete) on the dataset stored in a CSV file.
  - Automatically retrain the sentiment analysis model after dataset modifications to ensure up-to-date predictions.
- **Real-Time Interaction:** Provide a console-based UI for an interactive and user-friendly experience.

Example: Sentiment Analyzer with Dynamic Dataset Management

Required Python Modules:

1. NLTK: For natural language preprocessing tasks like tokenization, lemmatization, and stop-word removal. (**pip install nltk**)

2. scikit-learn: For feature extraction, building the Naive Bayes model, and evaluation metrics.

(**pip install scikit-learn**)

3. spaCy (Optional): If you want to extend or enhance text preprocessing using spaCy.

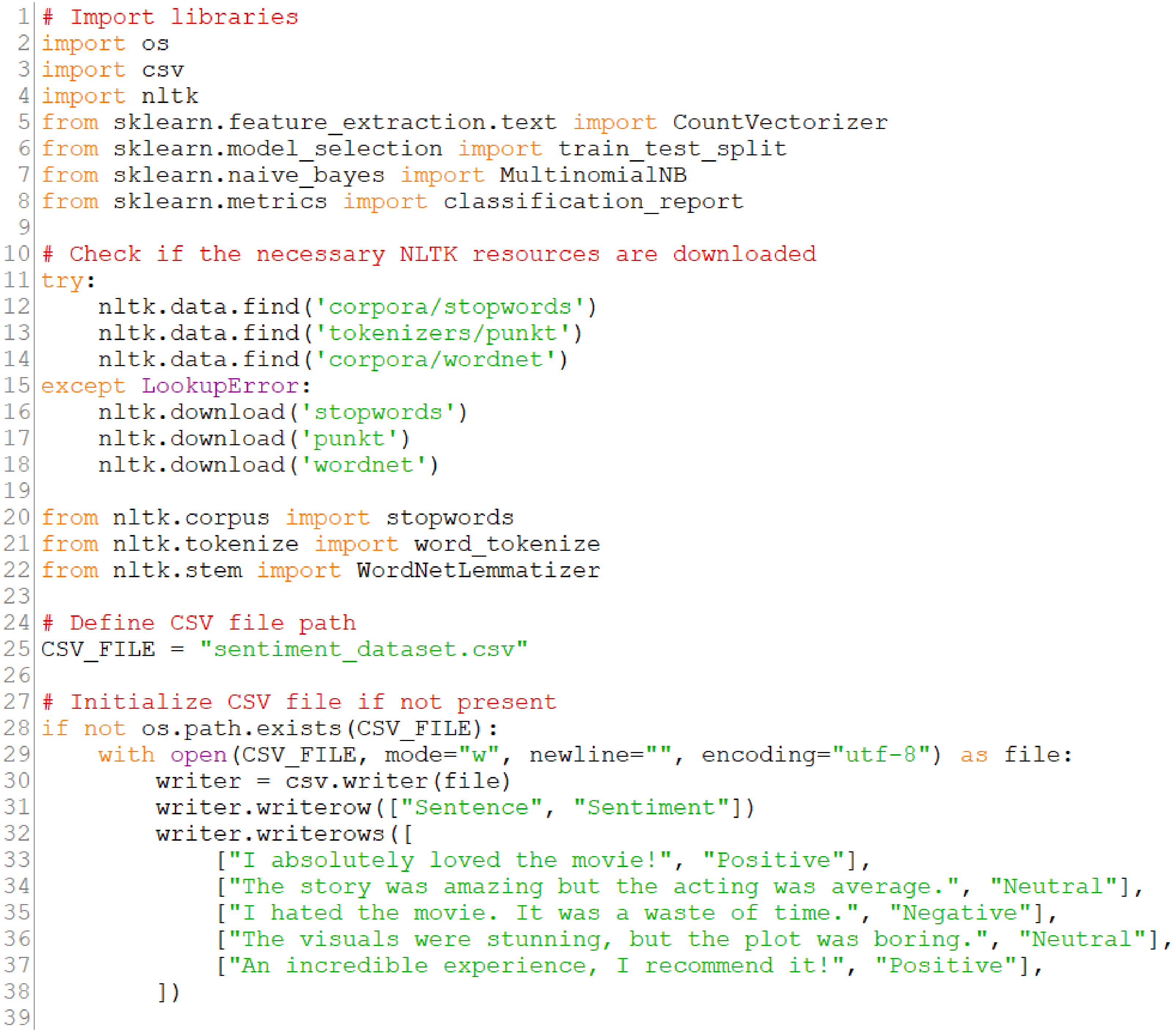
(**pip install spacy**)

4. Additionally, download a language model for spaCy (if you decide to use it):

(**python -m spacy download en_core_web_sm**)

Example:

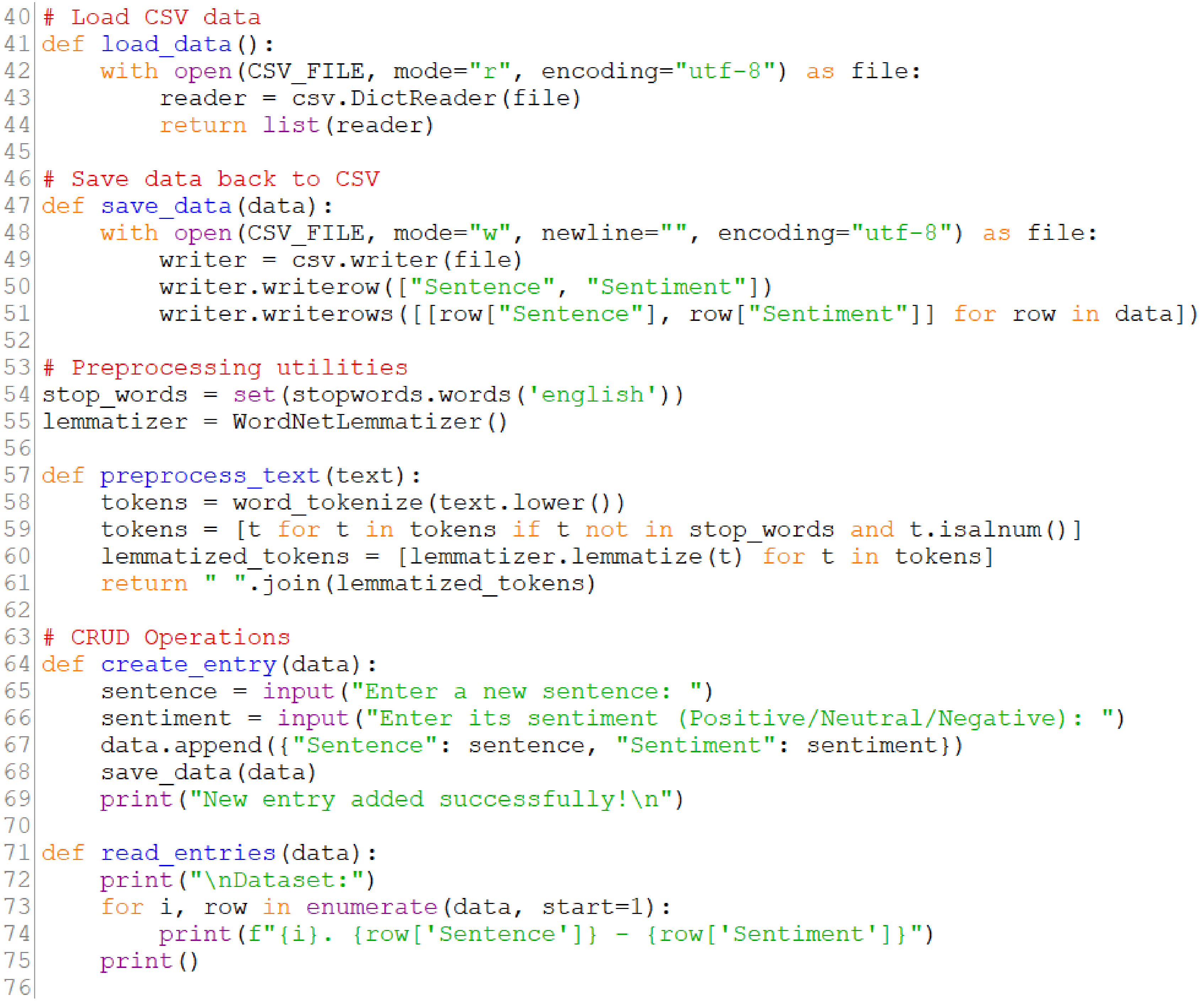
Sentiment Analyzer with Dynamic Dataset Management



```python
# Import libraries
import os
import csv
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report

# Check if the necessary NLTK resources are downloaded
try:
    nltk.data.find('corpora/stopwords')
    nltk.data.find('tokenizers/punkt')
    nltk.data.find('corpora/wordnet')
except LookupError:
    nltk.download('stopwords')
    nltk.download('punkt')
    nltk.download('wordnet')

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# Define CSV file path
CSV_FILE = "sentiment_dataset.csv"

# Initialize CSV file if not present
if not os.path.exists(CSV_FILE):
    with open(CSV_FILE, mode="w", newline="", encoding="utf-8") as file:
        writer = csv.writer(file)
        writer.writerow(["Sentence", "Sentiment"])
        writer.writerows([
            ["I absolutely loved the movie!", "Positive"],
            ["The story was amazing but the acting was average.", "Neutral"],
            ["I hated the movie. It was a waste of time.", "Negative"],
            ["The visuals were stunning, but the plot was boring.", "Neutral"],
            ["An incredible experience, I recommend it!", "Positive"],
        ])

```

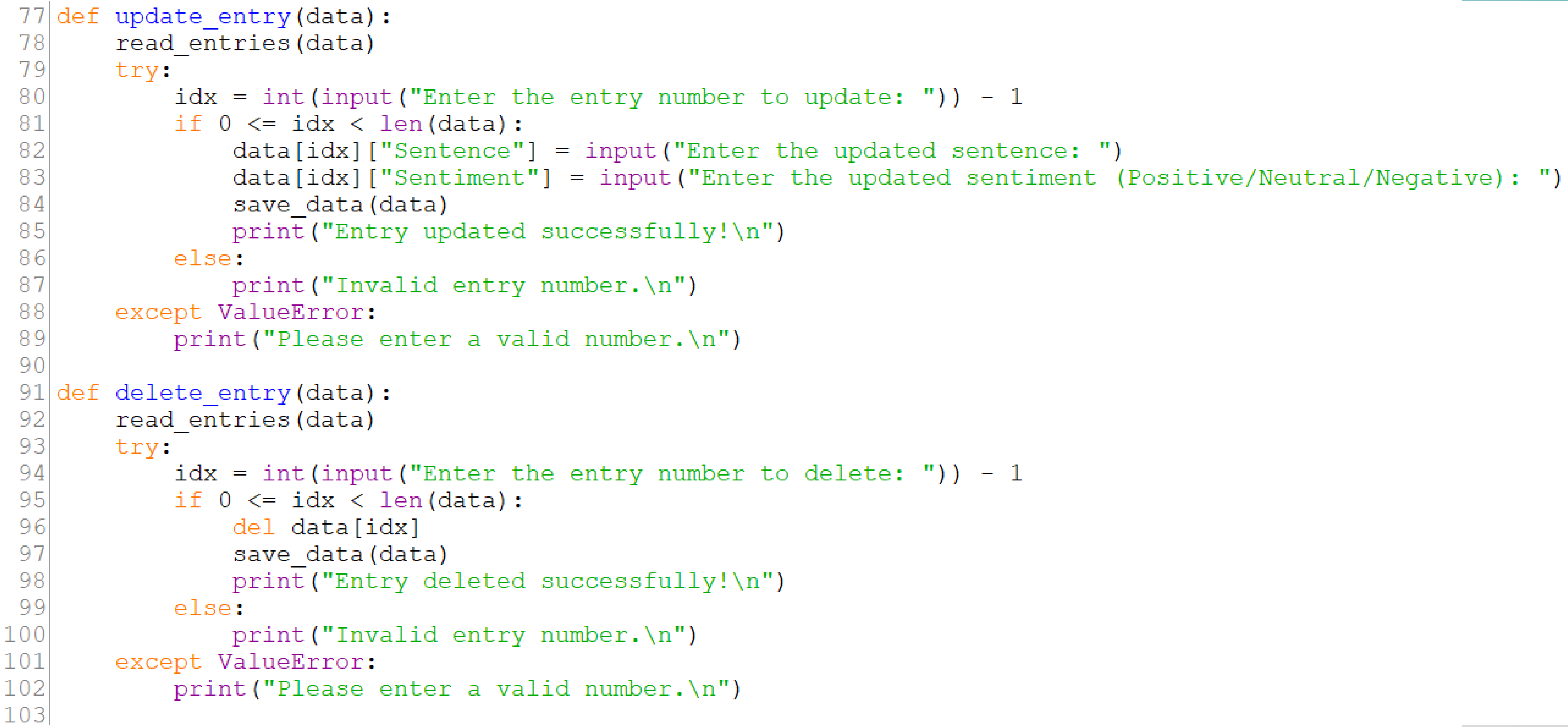
Example:

Sentiment Analyzer with Dynamic Dataset Management



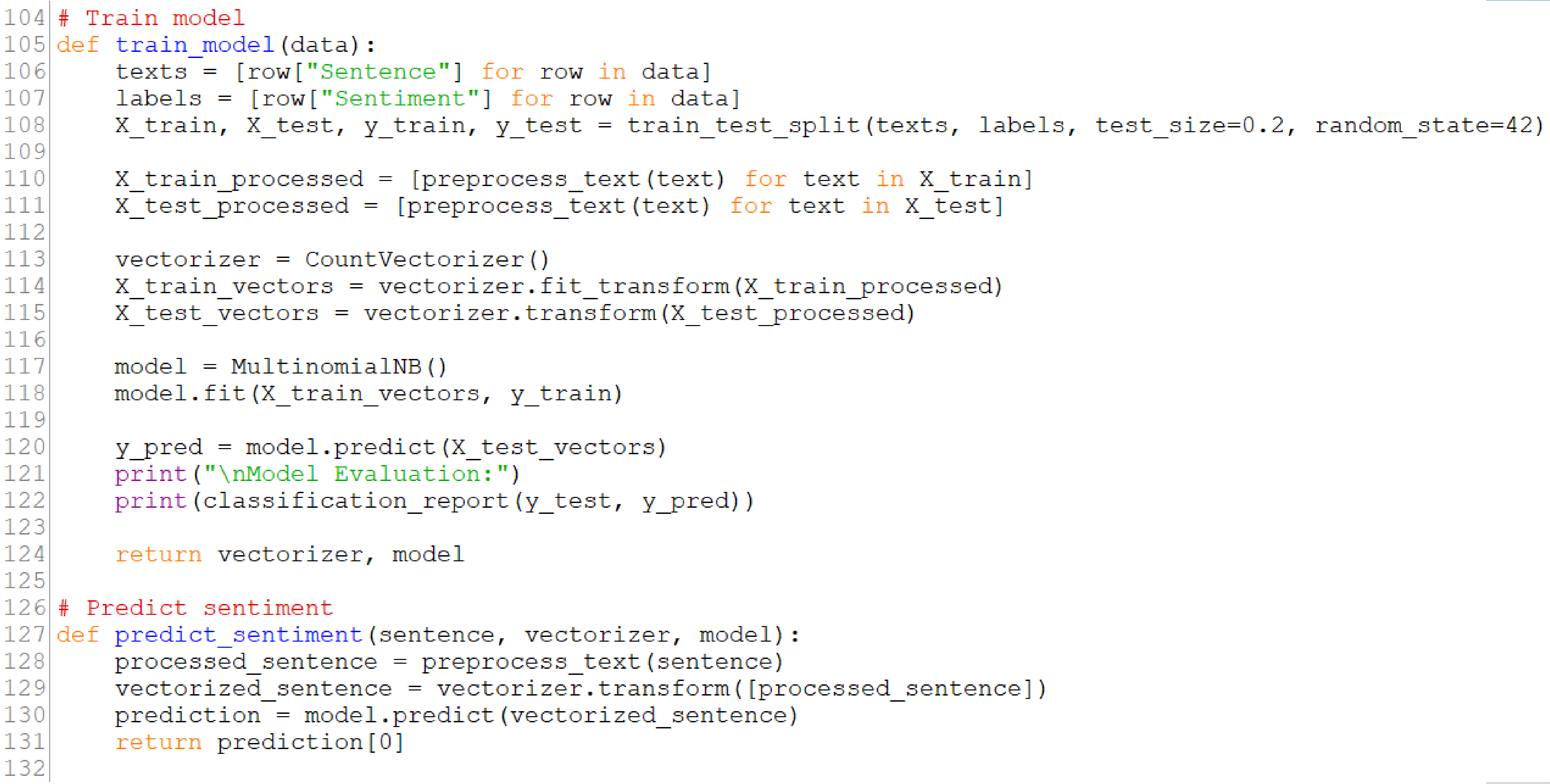
```python
# Load CSV data
def load_data():
    with open(CSV_FILE, mode="r", encoding="utf-8") as file:
        reader = csv.DictReader(file)
        return list(reader)

# Save data back to CSV
def save_data(data):
    with open(CSV_FILE, mode="w", newline="", encoding="utf-8") as file:
        writer = csv.writer(file)
        writer.writerow(["Sentence", "Sentiment"])
        writer.writerows([[row["Sentence"], row["Sentiment"]] for row in data])

# Preprocessing utilities
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    tokens = word_tokenize(text.lower())
    tokens = [t for t in tokens if t not in stop_words and t.isalnum()]
    lemmatized_tokens = [lemmatizer.lemmatize(t) for t in tokens]
    return " ".join(lemmatized_tokens)

# CRUD Operations
def create_entry(data):
    sentence = input("Enter a new sentence: ")
    sentiment = input("Enter its sentiment (Positive/Neutral/Negative): ")
    data.append({"Sentence": sentence, "Sentiment": sentiment})
    save_data(data)
    print("New entry added successfully!\n")

def read_entries(data):
    print("\nDataset:")
    for i, row in enumerate(data, start=1):
        print(f"{i}. {row['Sentence']} - {row['Sentiment']}")
    print()

```

Example: Sentiment Analyzer with Dynamic Dataset Management



```python
def update_entry(data):
    read_entries(data)
    try:
        idx = int(input("Enter the entry number to update: ")) - 1
        if 0 <= idx < len(data):
            data[idx]["Sentence"] = input("Enter the updated sentence: ")
            data[idx]["Sentiment"] = input("Enter the updated sentiment (Positive/Neutral/Negative): ")
            save_data(data)
            print("Entry updated successfully!\n")
        else:
            print("Invalid entry number.\n")
    except ValueError:
        print("Please enter a valid number.\n")

def delete_entry(data):
    read_entries(data)
    try:
        idx = int(input("Enter the entry number to delete: ")) - 1
        if 0 <= idx < len(data):
            del data[idx]
            save_data(data)
            print("Entry deleted successfully!\n")
        else:
            print("Invalid entry number.\n")
    except ValueError:
        print("Please enter a valid number.\n")

```

Example: Sentiment Analyzer with Dynamic Dataset Management

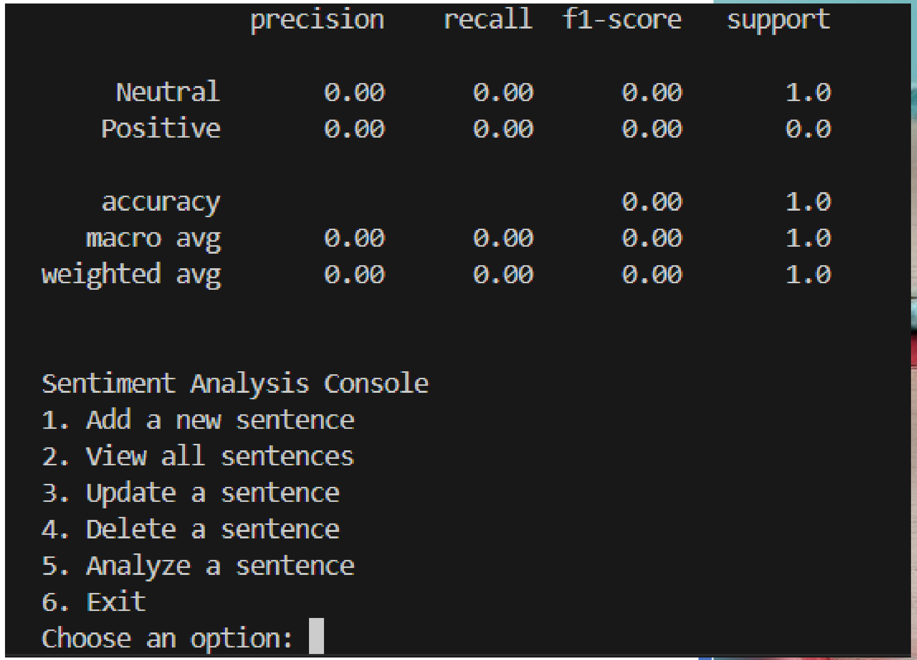


```python
# Train model
def train_model(data):
    texts = [row["Sentence"] for row in data]
    labels = [row["Sentiment"] for row in data]
    X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.2, random_state=42)

    X_train_processed = [preprocess_text(text) for text in X_train]
    X_test_processed = [preprocess_text(text) for text in X_test]

    vectorizer = CountVectorizer()
    X_train_vectors = vectorizer.fit_transform(X_train_processed)
    X_test_vectors = vectorizer.transform(X_test_processed)

    model = MultinomialNB()
    model.fit(X_train_vectors, y_train)

    y_pred = model.predict(X_test_vectors)
    print("\nModel Evaluation:")
    print(classification_report(y_test, y_pred))

    return vectorizer, model

# Predict sentiment
def predict_sentiment(sentence, vectorizer, model):
    processed_sentence = preprocess_text(sentence)
    vectorized_sentence = vectorizer.transform([processed_sentence])
    prediction = model.predict(vectorized_sentence)
    return prediction[0]

```

Example:

Sentiment Analyzer with Dynamic Dataset Management

```python
# Console UI
def main():
    data = load_data()
    vectorizer, model = train_model(data)

    while True:
        print("\nSentiment Analysis Console")
        print("1. Add a new sentence")
        print("2. View all sentences")
        print("3. Update a sentence")
        print("4. Delete a sentence")
        print("5. Analyze a sentence")
        print("6. Exit")
        choice = input("Choose an option: ")

        if choice == "1":
            create_entry(data)
            vectorizer, model = train_model(data)
        elif choice == "2":
            read_entries(data)
        elif choice == "3":
            update_entry(data)
            vectorizer, model = train_model(data)
        elif choice == "4":
            delete_entry(data)
            vectorizer, model = train_model(data)
        elif choice == "5":
            sentence = input("Enter a sentence to analyze: ")
            sentiment = predict_sentiment(sentence, vectorizer, model)
            print(f"Predicted Sentiment: {sentiment}\n")
        elif choice == "6":
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please try again.\n")

if __name__ == "__main__":
    main()
```



```
              precision    recall  f1-score   support

     Neutral       0.00      0.00      0.00       1.0
    Positive       0.00      0.00      0.00       0.0

    accuracy                           0.00       1.0
   macro avg       0.00      0.00      0.00       1.0
weighted avg       0.00      0.00      0.00       1.0


Sentiment Analysis Console
1. Add a new sentence
2. View all sentences
3. Update a sentence
4. Delete a sentence
5. Analyze a sentence
6. Exit
Choose an option: 
```

4. Homework

1. Introduction to NLP

- What is Natural Language Processing (NLP), and how is it applied in intelligence systems?
- List and explain two key NLP tasks required for intelligence systems.

2. Integrating NLP

- How does NLP improve the decision-making capabilities of intelligence systems?
- Describe the architecture of an NLP-integrated intelligence system and its key components.

សាកលវិទ្យាល័យន័រតុន
NORTON UNIVERSITY

Thank you