Language Detection Chatbot Web Application

M.Junaid 109(4B)

Table of Contents

1. Introduction

This project is a web-based chatbot that detects the language of user messages using the LangDetect library. It is built with Flask and supports basic natural language processing (NLP) features such as tokenization and stopword removal using NLTK. It logs the detected language and chatbot responses into a CSV file for future analysis.

2. Objective

The main goal of this project is to identify the language of input text entered by a user and provide an appropriate response in return. This functionality is useful in multilingual applications where automatic language identification is required.

3. Libraries Used

- flask: For building the web application backend.
- langdetect: For detecting the language of the user input.
- nltk: For text processing, tokenization, and stopword removal.
- csv, os, datetime: For logging chat interactions in a structured format.
- pandas : For analyzing the logged chat data.

```
app.py
from flask import Flask, render_template, request, jsonify
from langdetect import detect, DetectorFactory
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import pandas as pd
from datetime import datetime
import csv
import os
```

4. Features

• Accepts user messages through a web interface.
• Detects the language of the message using LangDetect.
• Cleans and tokenizes the message using NLTK.
• Returns an appropriate language-specific response.
• Logs all messages with timestamp, detected language, and response in a CSV file.

5. Folder Structure

• app.py → Main Flask application file
• templates/index.html → HTML front-end
• data/chat_logs.csv → File to store all chat logs

6. Flask Routes

• '/' → Renders the home page (index.html).

```
@app.route('/')
def home():
    return render_template('index.html')
```

• '/detect' → POST API endpoint that receives user input and returns detected language, response, and processed tokens.

```python
@app.route('/detect', methods=['POST'])
def detect_language():
    data = request.get_json()
    message = data.get('message', '')

    if not message or len(message.strip()) < 3:
        return jsonify({'error': 'Message too short for language detection'}), 400

    try:

        detected_lang = detect(message)
        print(f"Debug: Message: '{message}', Detected Language: {detected_lang}")
        processed_tokens = process_text(message)
        response = get_response(detected_lang)
        with open('data/chat_logs.csv', 'a', newline='', encoding='utf-8') as f:
            writer = csv.writer(f)
            writer.writerow([datetime.now().strftime('%Y-%m-%d %H:%M:%S'),
                             message, detected_lang, response])
        return jsonify({
            'language': detected_lang,
            'response': response,
            'processed_tokens': processed_tokens
        })

    except Exception as e:
        print(f"Error in detection: {e}")
        return jsonify({'error': f"Language detection failed: {str(e)}"}), 500
```

7. Detailed Code Description

• `process_text(text)` - Tokenizes the message, converts to lowercase, removes punctuation and stopwords.

```python
def process_text(text):
    try:

        tokens = word_tokenize(text)

        stop_words = set(stopwords.words('english'))
        tokens = [token.lower() for token in tokens if token.isalpha() and token.lower() not in stop_words]
        return tokens
    except Exception as e:
        print(f"Error processing text: {e}")
        return []
```

• `get_response(lang)` - Returns predefined chatbot response based on detected language.

```python
def get_response(lang):
    responses = {
        'en': "I detected English! ",
        'es': "¡Detecté español!",
        'fr': "J'ai détecté le français ! ",
        'de': "Ich habe Deutsch erkannt!",
        'it': "Ho rilevato l'italiano! ",
        'pt': "Detectei português! ",
        'ur': "میں نے اردو کا پتہ لگایا! ",
        'ar': "لقد اكتشفت اللغة العربية",
        'default': "I detected a different language!"
    }
    return responses.get(lang, responses['default'])
```

• `detect_language()` - Main logic to receive input, detect language, process it, generate response, and log the conversation.

```python
def detect_language():
    data = request.get_json()
    message = data.get('message', '')

    if not message or len(message.strip()) < 3:
        return jsonify({'error': 'Message too short for language detection'}), 40

    try:

        detected_lang = detect(message)
        print(f"Debug: Message: '{message}', Detected Language: {detected_lang}")
        processed_tokens = process_text(message)
        response = get_response(detected_lang)
        with open('data/chat_logs.csv', 'a', newline='', encoding='utf-8') as f:
            writer = csv.writer(f)
            writer.writerow([datetime.now().strftime('%Y-%m-%d %H:%M:%S'),
                             message, detected_lang, response])
        return jsonify({
            'language': detected_lang,
            'response': response,
            'processed_tokens': processed_tokens
        })

    except Exception as e:
        print(f"Error in detection: {e}")
        return jsonify({'error': f"Language detection failed: {str(e)}"}), 500
```

• `chat_logs.csv` - Stores timestamped records of all user queries and chatbot responses.

```
os.makedirs('data', exist_ok=True)
if not os.path.exists('data/chat_logs.csv'):
    with open('data/chat_logs.csv', 'w', newline='', encoding='utf-8') as f:
        writer = csv.writer(f)
        writer.writerow(['timestamp', 'user_message', 'detected_language', 'bot_response'])
```

8. Output