"MedChatBot: A Symptom-Based Medicine Recommendation System"

Group Information,

Name	ID	Contribution
Fahmida Anjum	2021 – 2 – 60 – 027	25%
Nirzona Binta Badal	2021 – 2 – 60 – 051	25%
Tithi Paul	2021 – 2 – 60 – 057	25%
Ismail Mahmud Nur	2021 – 2 – 60 – 052	25%

Submitted to,

Mohammad Rifat Ahmmad Rashid
Associate Professor
Department of Computer Science and Engineering
East-West University

Course Information,

CSE366
Artificial Intelligence
Section-03

Date of Submission: 29th May 2024

Abstract

MedChatBot is an innovative chatbot designed to recommend medicines based on user-provided symptoms. Utilizing natural language processing (NLP) with spaCy, the chatbot interprets symptoms, expands them using a synonym dictionary, and matches them against a comprehensive dataset of medicines. This report details the project's objectives, methodology, results, discussion, advantages, and potential applications.

Introduction

Efficient identification and recommendation of symptom-based medicines can significantly improve healthcare outcomes. MedChatBot aims to streamline this process by employing NLP to understand user symptoms and suggest appropriate medication from a comprehensive dataset. This tool enhances accessibility to healthcare information, making it easier for users to find suitable medication and seek timely medical advice.

Methodology

Data and Tools

- **Dataset:** comprehensive_medicine_list_with_symptoms.csv
- **Tools:**
  - **Streamlit:** For the user interface
  - **spaCy:** For NLP processing
  - **Pandas:** For data manipulation

Process Overview

1. **Data Loading:** Import the medicine dataset into a Pandas DataFrame.
2. **NLP Model:** Load spaCy's en_core_web_sm model.
3. **Symptom Preprocessing:**
   - Convert symptoms to lowercase.
   - Remove stop words and punctuation.
   - Extract lemmatized tokens and noun chunks.
4. **Symptom Expansion:** Use a predefined dictionary to expand symptoms with synonyms.
5. **Medicine Matching:**

- Process user input to extract and expand symptoms.
- Match expanded symptoms against the dataset.
- Return relevant medicine details.

Key Code Snippets:-

Data Loading and NLP Model:

```
import pandas as pd
import streamlit as st
import spacy

# Load the dataset
file_path = 'comprehensive_medicine_list_with_symptoms.csv'
medicine_data = pd.read_csv(file_path)

# Load spaCy model
nlp = spacy.load('en_core_web_sm')
```

Symptom Preprocessing:

```
# Function to preprocess the symptoms text using spaCy
def preprocess_text(text):
  doc = nlp(text.lower())  # Convert text to lowercase and process with spaCy
  tokens = [token.lemma_ for token in doc if not token.is_stop and not token.is_punct]
  phrases = [chunk.text for chunk in doc.noun_chunks]  # Extract noun phrases
  return tokens + phrases  # Combine tokens and noun phrases
```

Symptom Expansion:

```
# Function to expand symptoms with synonyms
def expand_with_synonyms(symptoms, synonym_dict):
    expanded_symptoms = symptoms.copy()  # Copy original symptoms
    for symptom in symptoms:
        for key, values in synonym_dict.items():
            if symptom in values or symptom == key:
                expanded_symptoms.append(key)  # Add the key (main symptom)
                expanded_symptoms.extend(values)  # Add all synonyms
    return list(set(expanded_symptoms))  # Remove duplicates and return as a list
```

Medicine Matching:

```
def find_medicines(symptom, data, synonym_dict):
    processed_symptom = preprocess_text(symptom)
    expanded_symptom = expand_with_synonyms(processed_symptom,
                                            synonym_dict)
    matches = pd.DataFrame()
    for word in expanded_symptom:
        word_matches = data[data['Indications (Symptoms)'].apply(
            lambda x: isinstance(x, str) and word in x)]
        matches = pd.concat([matches, word_matches])
    if not matches.empty:
        matches = matches.drop_duplicates(subset=['Brand Name',
                                                  'Medicine Type',
                                                  'Strength',
                                                  'Generic Name',
                                                  'Drug Class',
                                                  'Manufacturer'])
    return matches[['Brand Name', 'Medicine Type', 'Strength',
                    'Generic Name', 'Drug Class', 'Manufacturer',
                    'Indications (Symptoms)']]
```

User Interface with Streamlit:

```
# Initialize session state
if 'history' not in st.session_state:
    st.session_state.history = []  # Initialize chat history in session state

if 'temp_input' not in st.session_state:
    st.session_state.temp_input = ""  # Initialize temporary input in session state

def med_chatbot():
  st.title("MedChatBot")  # Set the title of the Streamlit app

  with st.sidebar:
    st.header("Common Symptoms")
    if st.button("Fever and Cough"):
      st.session_state.temp_input = "Fever, Cough" # Pre-defined input for Fever,Cough
    if st.button("Headache"):
      st.session_state.temp_input = "Headache" # Set pre-defined input for Headache
    if st.button("Itching"):
      st.session_state.temp_input = "Itching" # Set pre-defined input for Itching
    if st.button("Skin Rash"):
      st.session_state.temp_input = "Rash"  # Set pre-defined input for Rash
```

```
    # Temporary variable to hold user input
    temp_user_input = st.text_input("Describe your symptoms:", value=st.session_state
    .temp_input, key='user_input')  # Get user input with pre-defined values if any

    # Clear history button
    if st.button("Clear History"):
        st.session_state.history = []  # Clear chat history
        st.session_state.temp_input = ""  # Clear temporary input
        st.experimental_rerun()  # Rerun the script to reload the page

    # Process user input
    if temp_user_input:
        st.session_state.history.append({"type": "user", "message": temp_user_input})
        results = find_medicines(temp_user_input, medicine_data, synonyms)
        if not results.empty:
            st.session_state.history.append({"type": "bot", "message": results})
        else:
            response = "No medicines found for the given symptoms.
            Please try describing your symptoms differently or more specifically."
            st.session_state.history.append({"type": "bot", "message": response})

        # Clear input after submission
        st.session_state.temp_input = ""  # Clear temporary input after processing

    # Display chat history
    for chat in st.session_state.history:
       if chat["type"] == "user":
         st.markdown(f"**You:** {chat['message']}")  # Display user message
       else:
         if isinstance(chat['message'], pd.DataFrame):
           st.markdown(f"**MedChatBot:** Here are some medicines that might help:")
           st.dataframe(chat['message']) # Display DataFrame of medicines
         else:
           st.markdown(f"**MedChatBot:** {chat['message']}") # Display bot response

if __name__ == "__main__":
    med_chatbot()  # Run the med_chatbot function
```

Results

MedChatBot successfully processes user inputs to identify symptoms and recommends appropriate medicines. The use of synonym expansion enhances the accuracy of matches, ensuring relevant suggestions.

Example Interaction

User Input: "Fever, Pain"
Bot Response: Displays a table of medicines suitable for fever and pain, including their brand names, types, strengths, and manufacturers.

User Input: "Headache"
Bot Response: Displays a table of medicines suitable for Headaches, including their brand names, types, strengths, and manufacturers.

Discussion

MedChatBot demonstrates the practical application of NLP in healthcare. By expanding symptom inputs with synonyms, the system can better match user descriptions to relevant medicines, accommodating various terminologies. The user-friendly Streamlit interface ensures accessibility for non-technical users. This system can potentially reduce the time spent searching for suitable over-the-counter medications and can serve as a preliminary step before consulting a healthcare provider.

Advantages

- **Accessibility:** Provides quick and easy access to medication recommendations based on symptoms.
- **Efficiency:** Reduces the time and effort required to find suitable medicines.
- **NLP Integration:** Utilizes advanced NLP techniques to accurately interpret user inputs.
- **User-Friendly Interface:** Streamlit ensures that the application is easy to use, even for non-technical users.
- **Comprehensive Database:** Leverages a detailed dataset of medicines to provide accurate recommendations.
- **Educational Tool:** Can be used to educate users about different medications and their uses.

Conclusion

MedChatBot is an effective solution for symptom-based medicine recommendations, showcasing the potential of NLP in healthcare. This project has significant scope for further development and publication in academic and industrial domains. Future enhancements could include expanding the dataset, integrating with electronic health records, and adding more advanced NLP features.

References

- spaCy NLP Library: https://spacy.io/
- Streamlit: https://streamlit.io/
- Comprehensive Medicine Dataset: comprehensive_medicine_list_with_symptoms.csv
- https://www.kaggle.com/datasets/ahmedshahriarsakib/assorted-medicine-dataset-of-bangladesh?resource