Republic of Tunisia

Ministry of Higher Education and Scientific Research

5th Grade Cycle of Software Engineering

NLP-Based Disease Prediction System

Directed by:
Talel Haddeji
Ahmed Ben Amira

Professor:
Mr. Nizar Omheni

Academic Year 2024-2025

Contents

List of Figures

Abstract

This project aims to develop an NLP-based disease prediction system that takes user-inputted symptoms, normalizes their syntax, predicts the most probable disease, and provides necessary precautions. By leveraging machine learning and natural language processing, the system ensures accuracy and usability across multiple languages. Implemented using Flask for the backend and React for the frontend, the tool delivers predictions and guidance seamlessly, supporting users in English, French, and Arabic.

CHAPTER 1

Introduction

Contents

Introduction

1.1 Background

The early diagnosis of diseases significantly improves treatment outcomes. However, interpreting symptoms and mapping them to diseases can be complex, particularly in multilingual settings. A tool that automates this process can bridge the gap, enhancing healthcare accessibility.

1.2 Problem Statement

Patients often describe symptoms ambiguously or in different languages. Mapping these inputs to accurate diseases requires both normalization and multilingual handling. Manual processes are error-prone and inefficient.

1.3 Objective

This project addresses these challenges by:

1. Normalizing symptom descriptions using NLP techniques.
2. Predicting diseases with high accuracy.
3. Supporting multilingual input/output for user convenience.
4. Suggesting precautionary measures for the predicted disease.

CHAPTER 2

Dataset Overview

Contents

Dataset Overview

2.1 Source and Description

The data used in this project was sourced from Kaggle, specifically:

1. **DiseaseAndSymptoms.csv:** This dataset maps diseases to multiple associated symptoms.
   Each row represents a disease, and the columns correspond to various symptoms.

2. **Disease_precaution.csv:** This dataset provides precautionary measures for diseases. Each row contains a disease name and up to four suggested precautions.

2.2 Data Cleaning and Preparation

Data cleaning and preprocessing were essential to ensure the datasets were usable and consistent:

- **Handling Missing Values:** Rows with null or incomplete symptom data were removed.
- **Normalization:** Disease and symptom names were converted to lowercase to maintain uniformity.
- **Filtering Irrelevant Data:** Only diseases with at least one associated precaution were retained.
- **Finalized Datasets:**
  - **diseases.csv:** A cleaned version of DiseaseAndSymptoms.csv, structured to align with the model's requirements.
  - **Disease_precaution.csv:** Maintained its original structure but was cross-referenced to ensure alignment with diseases.csv.

By the end of preprocessing, the datasets were ready for use in disease prediction and precaution retrieval.

CHAPTER 3

Methodology

Contents

Methodology

3.1 Workflow

The workflow for the system is designed to integrate natural language processing and machine learning techniques with a user-friendly interface:

1. **Input Symptoms:** Users enter symptoms as a comma-separated string. Inputs can be in English, French, or Arabic.
2. **Translation:**If symptoms are entered in French or Arabic, they are translated to English using Google Translate API.
3. **Symptom Normalization:**Symptoms are matched against a pre-defined list using fuzzy string matching, ensuring spelling corrections and synonym handling.

Disease Prediction Using a TF-IDF-based vectorization of symptoms and cosine similarity, the most probable disease is identified.

Output Results: Predicted disease and precautions are optionally translated back to the user's preferred language.

Symptom Normalization: Symptoms are matched against a pre-defined list using fuzzy string matching, ensuring spelling corrections and synonym handling.

3.2 Backend Model

The backend, built using Flask, implements the core functionalities:

- Normalization of symptoms.
- Disease prediction using cosine similarity.
- Multilingual support through Google Translate API.

The DiseasePredictor class in $disease_model.pyencapsulatesthesefeaturesformodularandreusablecod$

3.3 Frontend

The frontend, developed in React, provides:

- A simple input form for symptoms and language preference.
- A dynamic display of the predicted disease and precautions.
- Multilingual output for enhanced user experience.

CHAPTER 4

Implementation

Contents

Implementation

4.1 Symptom Normalization

Normalization is critical to handle variations in symptom inputs:

- **Exact Match:** Symptoms matching directly with the predefined list are accepted.
- **Fuzzy Match:** For inputs with minor typos or variations, thefuzz library is used to find the closest match (e.g., "fver" → "fever").
- **Threshold:** Matches are accepted if the similarity score exceeds 80 %.

4.2 Disease Prediction

The prediction model uses the following approach:

- **TF-IDF Vectorizer:** Converts symptom sets into numerical vectors by calculating term frequency-inverse document frequency.
- **Cosine Similarity:** Measures the similarity between the user's symptoms and pre-existing disease symptom vectors.
- **Result:** The disease with the highest similarity score is returned as the prediction.hes are accepted if the similarity score exceeds 80 %.

4.3 Multilingual Support

To handle multilingual inputs:

- **Google Translate API:** Translates symptoms from French or Arabic into English for prediction.
- **Output Translation:** The predicted disease and precautions are translated back to the user's selected language if it's not English.

CHAPTER 5

Testing and Validationn

Contents

Testing and Validation

5.1 Test Cases

Extensive testing was conducted with diverse inputs:

- Correctly formatted symptom inputs in English.
- Misspelled or incomplete symptoms.
- Symptom inputs in French and Arabic.

5.2 Metrics

- **Accuracy:** Measured the correctness of disease predictions.
- **Precision:** Evaluated the relevance of precautions provided.
- **Translation Precision:** Assessed the accuracy of symptom and result translations.

5.3 Results

- **Prediction Accuracy:** Approximately 85 %, based on validation with known mappings.
- **Fuzzy Matching Success Rate:** 90 % for minor misspellings and synonym variations.
- **Translation Precision:** 90 % accurate for medical terms in French and Arabic

CHAPTER 6

Challenges and Solutions

Contents

Challenges and Solutions

6.1 Data Challenges

- **Issue:** Missing values in symptoms or precautions.
- **Solution:** Data cleaning and preprocessing eliminated unusable rows.

6.2 Multilingual Translation

- **Issue:** Occasional inaccuracies in Google Translate for domain-specific terms.
- **Solution:** Manual cross-verification and fallback mechanisms.

6.3 Symptom Variations

- **Issue:** Synonyms or colloquial terms not directly matching the dataset.
- **Solution:** Expanded the symptom database and used fuzzy matching to handle variations effectively.

CHAPTER 7

Conclusion and Future Work

Contents

Conclusion and Future Work

7.1 Conclusion

The developed system achieves its goal of providing accurate and multilingual disease prediction based on user symptoms. By integrating NLP, machine learning, and a user-friendly interface, the tool bridges a crucial gap in early disease detection.

7.2 Future Work

- **Database Expansion:** Include more diseases and symptoms for broader applicability.
- **Severity Assessment:** Integrate severity levels for symptoms to refine predictions.
- **Advanced Models:** Experiment with transformer-based models (e.g., BERT) for symptom analysis.