Title: recommend relevant specializations (of doctors) to the patient.

Introduction:

The goal of this project is to design and implement a system that asks patients their symptoms, and related questions, predict the possible conditions and recommend relevant specializations (of doctors) to the patient.. The system uses Noisy OR Model that uses EMR(Electronic Medical Record) as database to build a knowledge graph which will map symptoms to diseases. As the web servers like google are used by the people to search for any disease whenever they are having symptoms like cold , fever or stomach pain, implementation of an automated system is very beneficial.

Solution:

First, gather a large dataset of patient symptoms, possible conditions, and relevant specializations of doctors. This dataset would be used to train Noisy OR model.

For training the model, first extract the disease and symptom from the training data(unstructured from EMR). For training data we can use medical records from various hospitals. Second, statistical models relating diseases and symptoms are learned. Third, the learned statistical models were translated into knowledge graphs.

Next, use NLP techniques to process the patient's input and extract relevant information such as symptoms and other related questions. This information would then beinput into the trained machine learning model, which would make predictions about possible conditions and relevant specializations of doctors.

Structure of Dataset:

The structure of the dataset for this machine learning model would include features such as patient demographic information, symptom information, and relevant medical history.The dataset would also need to include information on the different medical conditions and the relevant specializations of doctors for each condition.

Here is an example of what a sample data might look like:

Patient ID	Age	Gender	Symptom 1	Symptom 2	Symptom 3	Medical History	Condition	Specialization
1	45	Female	Headache	Nausea	Vomiting	Migraine		Neurologist

Tools & Technologies:

Natural Language Processing (NLP) library: To process and understand the patients' symptoms and related questions, I would use a NLP library such as NLTK, spaCy, or Gensim. These libraries provide tools for tokenization, stemming, lemmatization, and other NLP tasks that can help make sense of the patient's input.

Machine Learning Library: To predict the possible conditions and specializations based on the patients' symptoms and related questions, I would use a machine learning library such as scikit-learn, TensorFlow, or Keras. These libraries provide a wide range of algorithms and tools for building and evaluating predictive models.

Database: To store and retrieve the patient's data and the trained models, I would use a database such as MySQL, MongoDB, or PostgreSQL.

Web framework: To build the web-based interface for the system, I would use a web framework such as Flask, Django, or Ruby on Rails. These frameworks provide tools for building web applications and handling user input.

Deployment platform: To deploy the system in production, I would use a cloud-based platform such as AWS, GCP, or Heroku.

The reason I would use these tools is that they are widely used and well-supported in the industry and have a large community of developers and users. They are also open-source and have a wide range of functionality and libraries that can be easily integrated with other tools.

The pros of these tools are that they are widely used and well-supported, have a large community of developers and users, and are open-source. The cons are that they may be more difficult to set up and configure than proprietary tools, and may not have as much support or documentation as proprietary tools.

Algorithms:

One possible approach to implement this system using a noisy or model would be as follows:

Collect a dataset of symptoms, related questions, and corresponding diagnoses.

Use the Noisy OR model to represent the relationships between symptoms and conditions. The Noisy OR model is a probabilistic graphical model that can represent the presence or absence of symptoms as binary variables and the presence or absence of a condition as a binary variable. The model can handle uncertainty and missing data.

Train the model on the dataset using the Expectation-Maximization (EM) algorithm or other techniques to estimate the parameters of the model.

Use the trained model to predict the probability of a condition given a set of symptoms and related questions for a new patient.

Use the predicted condition to recommend relevant specializations (of doctors) to the patient.

The noisy or model can handle uncertainty and missing data, which is important in this task since patients may not report all of their symptoms or may not be sure about the symptoms they are experiencing. The model can also handle a large number of input features and can be used to handle the potential for multiple diagnoses.

The pros of this approach include the ability to handle uncertainty and missing data, the ability to handle a large number of input features and the ability to handle the potential for multiple diagnoses. The cons include the need for large amounts of labeled data and the potential for overfitting if the model is too complex.

Alternate Algorithm :

K-Nearest Neighbors(KNN),Logistic Regression, Support Vector Machines (SVMs)

RNN are used for the unstructured data which can be used for EMR(Electronic Medical Records) for the data entered by the physicians and nurses for the patients.

Evaluation:

The model could be evaluated by verifying the disease-symptom relation by physicians and comparing it with the Google Health Knowledge Graph available.Accuracy of the model can be checked by dividing the number of correct predictions by the total number of predictions.

Recall can be use which measures the proportion of true positive predictions among all actual positive cases. It is calculated as the number of true positive predictions divided by the number of true positive predictions plus false negative predictions.

To make this a continuous learning system:

Regularly adding new data to the training dataset and retraining the model on the updated dataset, this will help the model to adapt to new and changing patterns in the data.Monitoring the performance of the model on unseen data and tuning the model if needed. Incorporating a feedback loop that allows the system to learn from its predictions and recommendations, and improve its performance over time.