

# Second Opinion, a Collaborative Online Game for Medical Diagnosis

Elena Agapie

Faculty mentor: Prof. Dr. Ken Goldberg

Graduate mentor: Ephrat Bitton

Automation Sciences Lab

Departments of IEOR and EECS

University of California, Berkeley

SUPERB 2008

## Abstract

Information technology has made tremendous impacts on the health care system. This work aims to develop an online intelligent diagnosis system, which will cooperate with human collaborators to improve its machine intelligence to provide fast and accurate diagnosis. We have developed an online interface so that a user can interact with the system as either a diagnoser or a diagnosee. As a diagnosee, the user can interact with the system to provide their symptoms. The system will use its intelligence to plan a strategy of questioning and finally provide a possible diagnosis. As a diagnoser, the user can review stored cases, plan his/her own strategy to interact with the system to learn symptoms of the selected case, and finally give his/her diagnosis. The system encourages the user to compete for points and for a good diagnoser ranking as a component of the game. Using such interaction with human collaborators, we have developed a machine learning algorithm so that the system can automatically improve its questioning strategy and provide faster and more reliable diagnosis. The system now focuses only on sexual transmitted diseases and will be extended to other diseases in future work.

## 1 Introduction

The advent of online systems that allow people to store their information in an easy and accessible way have enabled the creation of online collaborative tools that use large repositories of data. Medical diagnosis and treatment are areas of great importance that have yet to fully take advantage of such tools.

There are multiple advantages to opening automated diagnostic systems to the general public. It would facilitate an open, informed dialogue between patients and their health care providers, which will improve the quality of the care patients receive and reduce the number of diagnosis errors. A diagnosis system that is widely used and trusted by patients could help reduce the rising burden on the emergency departments, which have serious overcrowding problems [1].

People are turning to the Internet in increasing numbers to search for health information pertaining to their unique medical conditions to augment the services of primary care providers. If harnessed correctly, this information has the potential to serve as a useful tool for both patients and doctors alike. Studies have shown that some medical systems can accurately diagnose over 50 percent of real life cases. This gives reason to believe that an intelligent, personalized system can obtain greater diagnostic accuracy, based mainly on user-reported symptoms and medical history.

Motivated by these observations, our team is developing Second Opinion, an online game designed to build a knowledge base of medical information that can one day be used to build an adaptive diagnosis algorithm. In the game, we invite users to create instances of medical cases by providing medical information of their current state and by answering questions asked by the system. The users receive a diagnosis in real time and have the chance to correct it if needed. The second component of the game consists in the diagnosers trying to give accurate diagnoses. They can do this by reviewing the information from the cases and by asking and receiving answers to questions that the user answered to. The diagnosis given by the diagnoser is verified when either the patient confirms the true diagnosis or when a majority of players independently agree on the diagnosis.

The system trains on the data provided by the users. We are using a machine learning approach to provide diagnosis based on the information collected by the system. The available database is updated with every diagnosed case made available. The diagnosis provided by the system is based exclusively on an algorithm that learns from the previous cases. Secondly, the system will use the wisdom of crowds to provide more accurate diagnosis. We weight the opinion of every diagnoser based on their success rate and use their knowledge to provide faster and better diagnosis.

The Second Opinion knowledge base will continue to grow as more people play the game and will thus one day allow for more sophisticated diagnosis algorithms. We propose a game-based model to invoke the competitive nature of players and encourage continued participation.

## 2 Related Work

Several systems partially implement components from Second Opinion. Online medical diagnosis systems are easily accessible and sometimes come with a wide variety of diagnoses. All of these systems use a static approach of using their data, without using the cases they collect for future diagnosis. Other type of online applications use extensively the information collected from users. In this category we encounter game type of systems that use the wisdom of crowds to solve artificial intelligence problems. For the theoretical area of our system there are multiple machine learning algorithms that discuss some of the problems that we encounter. In this section we will discuss in more detail other systems and algorithms that have similar components to Second Opinion.

Multiple **medical diagnostic systems** currently exist **online** that allow patients to search for possible diagnoses that can account for their symptoms. One of the more prominent resources is provided by *WebMD*, also called *Symptom Checker* (<http://symptoms.webmd.com/>). Patients seeking information about symptoms they are experiencing are asked to (anonymously) specify their gender, age group, and zip code, and to then identify their symptoms by interactively clicking on regions of an image of a body and choosing from a drop-down list. For some symptoms, patients are also given the option to answer a set of questions to determine, for example, what activities make the symptoms worse and to learn more about the patient's profile. Symptom Checker then compiles a list of potential causes of the symptoms, including links to pages providing more information about each diseases or condition. *Easy Diagnosis* (<http://easydiagnosis.com>) has diagnosis modules for a narrow set of symptoms. After the user provides their information in similar manner, they are presented with a range of possible causes and their corresponding probabilities. Other decision systems for diagnosis are *Wrong Diagnosis* (<http://www.wrongdiagnosis.com>) or *Your Diagnosis* (<http://yourdiagnosis.com/>).

While these systems provide a useful but limited set of tools for narrowing down the possible causes of a patient's symptoms, they are unable to learn and improve from the information provided by the patients. As such, they provide a superficial level of reasoning similar to that of looking up symptoms in a book.

Unlike the medical diagnosis systems, othe online systems are aiming to build their performance from the human input. Some of these systems use the **wisdom of crowds** under the assumption that decisions obtained from groups are better that the ones made by individuals. One of the first online games to harness this concept to train an articially intelligent agent is the computer-based version of the game *20 Questions*, called *20Q* [2]. Players of the game secretly think of an object and then answer a series of 20 questions about the object. The program uses the information collected from the player to determine the next question to ask about the object and, once it has used up 20 questions, to guess what the object is. The success rate is of 80 percent and if the program is allowed to ask 5 more questions it jumps to 98 percent. The program uses an artificial neural network to learn from each game session. The neural network operates by maintaining a large matrix indexed by objects and questions. The score of cells is adjusted based on a score received from user's responses at each play of the game.

Recent interest in using *human intelligence to solve difficult AI problems* was inspired by the success of the popular *ESP* game [3]. The game provides a new and creative approach to the problem of image labeling. A player of the ESP game is anonymously paired up with another player and both are shown an image randomly selected from the Internet. The two players must enter words that describes the contents of the image and are said to agree on the label when both of them enter the same word. Then, a new image is shown.

*Peekaboos* is a similar game that uses human intelligence to solve the problem of image recognition by having users take turns selecting parts of an image to reveal to the other player, who then must guess the label of the object. [4] The goal of the project is to build an extensive database of images and the locations of the objects within them, which can then be used to train image recognition algorithms.

Researchers have attempted to build **machine learning systems** to solve the problem of **medical diagnosis**, and have seen limited success. *MYCIN* is one of the earliest systems, developed at Stanford in the 1970's. The

system uses a knowledge base of if-then rules to identify harmful bacteria and to determine the appropriate antibiotic treatment given a patient’s body weight. A majority of 70 percent of the therapies recommended by the system were acceptable, compared to 50 percent prescribed by other experts [5].

Currently machine learning is used in a wide variety of tasks for medical diagnosis. Research has been done for the implementation of systems that provide diagnosis based on symptoms. In [6] we found a system for the diagnosis of epilepsy that is built on the 20Q approach of guessing the diagnosis. Other systems have implemented a diagnosis of sexually transmitted diseases based on their symptoms. [7] The approach used a neural network to determine the diagnosis, and the test data revealed a high rate of success. Another system was tested on a palm handheld in a pediatric emergency department. [8] The system was developed for mobile emergency triage of acute abdominal pains and it used decision trees to successfully classify 85 percent of the urgent cases.

Machine learning algorithms used in medical diagnosis are commonly trained on correctly diagnosed data. [9] There are some well established algorithms that are successfully used in machine learning. We have chosen the approach of decision trees learning that uses information gain and system entropy to classify samples, since the diagnosee must be queried for data. [10] There are also approaches that use a high amount of data that is not classified at all, thus using both labeled and unlabeled data. [11]. One of the most challenging problems was how to use the wisdom of crowds for diagnosis. The problem consists in having unreliably classified samples with several diagnosis and choosing what the best diagnosis is. The most well known algorithm is the Adaptive Boosting [12]. The algorithm uses weights for the classified data. The weights are updated at each round such that weights of correctly classified samples are increased and weights of misclassified data are decreased.

### 3 Diagnosis using Machine Learning and Wisdom of Crowds

Our system relies on two aspects of learning. The first one is to provide an accurate diagnosis based on a set of training data. We are given a set of training samples consisting of symptoms and the associated diagnosis. A diagnosee enters a case which the system assist in constructing. The system asks questions to find the symptoms of the case. It seeks to ask as few questions as possible to determine the probable diagnosis.

The second aspect is to decide what is a good consensus between the diagnosis given by several users to estimate the true diagnosis of a case. In the second case we are given a set of diagnosis for one case given by several experts. We can access a computed success rate of the experts. We would like to get a consensus from their diagnosis that would provide us with the most likely diagnosis.

#### 3.1 Provide a New Diagnosis Based on Previous Cases

The system will use a decision tree learning algorithm to provide a diagnosis to a new case based on the previous cases from the system. We use the C4.5 algorithm to build a decision tree from the set of available cases with their final diagnosis. The algorithm is based on building a decision tree and revising the set of target diseases as the tree is built. It uses information gain in deciding the structure of the tree.

We will assume the following notations:

- $D$  represents the random variable of diseases
- $\{d_1, \dots, d_n\}$  represents the set of values that  $D$  can take.
- $\{S_1, \dots, S_m\}$  represents the set of random variables of symptoms.
- $(t_1, \dots, t_m)$  represents the test instance that we want to classify; it is a succession of booleans, for each of the  $m$  symptoms, the value is 1 if symptom is present and 0 otherwise.
- $D^k$  represents the set of diseases that are considered in the classification after the  $k$ -th level of building the decision tree.

We assume that the set of diseases and symptoms are static, and that that symptoms are independent random variables. We will consider that we have a number of  $n$  diseases and  $m$  symptoms. The symptoms are associated with questions in the user-system interaction. When trying to classify a new case, we will say that attribute  $S$  is present if the entry of the new case contains  $S$  as one of the symptoms.

The algorithm builds a decision tree based on the concept of information entropy as a measure of the uncertainty associated with the random variable  $D$ . At every level of the tree the algorithm chooses the attribute with the most information gain. Based on the value of the attribute in the test instance, the diseases set is divided accordingly, until we reach a level of the tree when the disease set has one element.

The algorithm uses entropy as a measure of the information contained by the disease variable in association with symptoms. A high entropy gives more uncertainty an the diagnosis. That is why the formula for computing the entropy will return smaller values when the probability of certain disease is larger. The more certainty we have about the diagnosis, the lower the value of the entropy is. The **entropy** of the discrete variable  $D$  is initially defined as:

$$H(D) = - \sum_{i=1}^n Pr(D = d_i) \log_2 Pr(D = d_i)$$

The expected **information gain** is obtained from the change in information entropy from a prior state to the state that takes new information from the variable  $S$ , that is the information taken from the presence of a new symptom  $S$ . The information gain represents the reduction in entropy of  $D$  obtained after learning a new symptom  $S$ . We are using information gain to determine the prefered sequence of symptoms we want to evaluate. Because the system asks questions about the symptoms of the diagnosee, the fastest way to determine the diagnosis is by asking the questions that give the most information gain. It is defined as:

$$IG(D; S) = H(D) - H(D|S)$$

The entropy of the discrete variable  $D$ , given the information from the attribute  $S$ , is defined as the sum of entropies of the disease set partition, based on the values of  $S$ , and weighted by the probability of  $S$ :

$$H(D|S) = \sum_{s \in val(S)} Pr(S = s) H(D|S = s),$$

where  $val(S)$  represents the values that attribute  $S$  can take, respectively the values assigned to the symptom, in our case 0 or 1.

After choosing the first  $k$  symptoms, the diseases set is reduced to the subset  $D^k$ . We choose the best  $(k + 1)$ -th symptom by choosing out of the  $m - k$  remained symptoms the one that gives the highest information gain. The information gain when choosing the  $k + 1$  attribute  $S_\alpha$  is given by:

$$IG(D^k; S_0, \dots, S_k, S_\alpha) = H(D^k|S_0, \dots, S_k) - H(D^k|S_0, \dots, S_k, S_\alpha)$$

The  $k + 1$  chosen symptom  $S_\alpha$  has the property:

$$\alpha = \operatorname{argmax}_{i \in \{k+1, \dots, m\}} IG(D^k; S_0, \dots, S_k, S_i).$$

The next step continues building the tree with the set of diseases

$$D^{k+1} = \{d \in D^k | S \text{ has value } t_{k+1} \text{ in instance } d\}.$$

The algorithm has the following input, output and can be formalized as follows:

*Input* Training data:  $T_1, \dots, T_k$  where  $T_i = (t_{i1}, \dots, t_{im}, d_i)$ , for all  $i \in \{1, \dots, k\}$ , and  $t_{ij}$  represents the  $j$ -th symptom for the  $i$ -th training case, and  $d_i$  represents the disease classification of the instance.  
 Test data:  $(t_{k+1,1}, \dots, t_{k+1,m})$

*Output* disease classification  $d_{k+1}$  of the test data.

1. Check if all samples belong to the same class or all the symptoms give the same information gain
2. For each symptom  $S_i$
3. Find the normalized information gain from splitting on  $S_i$
4. Let  $S_{best}$  be the attribute with the highest normalized information gain
5. Create a decision node that splits on  $S_{best}$
6. Recur on the sublist obtained by splitting on  $S_{best}$  by the value from the test sample  $t_{best}$  and add the nodes of the symptoms that haven't been considered

### 3.2 Learning from Unreliable Data (or How to Use the Wisdom of Crowds)

We consider the problem of aggregating multiple diagnoser opinions for classification, given the past performance of each individual. More specifically, we seek to determine how to weight the opinion of each diagnoser and determine the final consensus to achieve maximum accuracy. The inputs and outputs of the problem are characterized as follows:

	Set of diagnoses provided by the $p$ diagnosers of the current case $j$ : $\{\gamma_{j1}, \dots, \gamma_{jp}\}$ , where $\gamma_{ji}$ represents the vector $(0, 0, \dots, 1, 0, \dots, 0)$ that has 1 on the $i$ -th position and 0 everywhere else.
<i>Input</i>	Diagnoser success rate $\{w_1, \dots, w_p\}$
	Number of cases diagnosed correctly by each diagnoser $\{c_{\gamma_1}, \dots, c_{\gamma_n}\}$
	Number of cases diagnosed by each diagnoser $\{t_1, \dots, t_n\}$
<i>Output</i>	Consensus diagnosis $\bar{d}$ given by the $p$ diagnosers
	Adjusted weights for the diagnoser

We will make the following assumptions:

- The system assumes no prior knowledge on the expertise of the diagnosers.
- The diagnoser's expertise is constant in time.
- The system does not consider the timing when the diagnoses was done. At some point, the case instance is closed for evaluation and the system looks at the diagnoses and outputs the final prediction.

**Diagnosis Estimation by Consensus.** For a case, we rank the diagnoses that were given during the game play according to a weighted sum of the diagnosers' opinions. The weight of each players opinion reflects his or her past performance (success rate). We can define the current best estimate of the true diagnosis by selecting the disease with the highest rank.

Given the set of all diseases  $\{d_1, \dots, d_n\}$ , each diagnoser will vote with value 1 for one of the disease and with value 0 for the others. Each disease will receive a rank based on the rank of the diagnosers. Initially, all the diseases have a rank of zero. When a diagnoser  $i$  with weight  $w_i$  provides a vote for disease  $d_j$ , the rank of the disease changes by:  $d_j = d_j + w_i$ . We compute the rank  $d_k$  of each disease by a weighted sum of the diagnosers' vote, and then choose for the consensus  $\bar{d}$  the disease with maximum rank.

$$\bar{d}_k = \sum_{k=1}^p \gamma_{ik} * w_k$$

$$\bar{d} = \underset{k}{\operatorname{argmax}} d_k.$$

**Readjusting the Weights of the Diagnosers.** The weights of the diagnosers are computed in a very simple way for now. We are assuming the weight of a diagnoser is their success rate. We measure the success rate by taking the number of cases they diagnosed in accordance to the consensus, divided by the total number of cases they diagnosed, for case  $i$  and user  $k$ .

$$w_k = \frac{w_k * c_{\gamma_k} + \gamma_{ik}(\bar{d})}{t_k + 1}.$$

where  $\gamma_{ik}(\bar{d})$  is 1 if the user  $k$  voted for the diagnosis  $\bar{d}$  in case  $i$ , and 0 otherwise.

## 4 Simulator Results

The system has not yet been released to the public. Thus, because of lack of real data, we built a simulator that would generate data on which we can simulate the behavior of the system. The simulator has two components: generating the data and running simulations using the consensus and weighting functions previously described.

The first component is the generator of data. We are generating the following sets of data:

- Diseases - we generate a uniform distribution of diseases.
- Symptoms - we generate a uniform distribution of symptoms for each disease.

- Diagnosers - we generate a uniform distribution of diagnosers. Diagnosers are characterized by their success rate of diagnosing diseases. The success rate is constant throughout the whole game and all diagnosers have the same success rate.
- Cases - we generate cases corresponding to the distribution of diseases and based on the distribution of symptoms corresponding to each disease. We characterize a case by the symptoms that are present and use the term true diagnosis for the disease corresponding to the symptoms.
- Estimated Diagnosis - based on the success rate of a diagnoser, we randomly generate a correct or wrong diagnosis for the case that is currently being diagnosed.

We first generate the distribution of diseases, of symptoms per disease, and of success rate. We then generate the cases based on the distribution of diseases and symptoms. We don't generate any data that doesn't fit these distributions.

The simulator simulates the system by using:

1. internal functions that generate the consensus of the diagnosers and a function that adjusts the weights of the diagnosers after a case is closed.
2. the decision tree algorithm to learn from the generated data, provide diagnosis and simulate diagnoser's behaviour.

For all our simulations we made the following assumptions regarding our parameters:

- the set of disease has 10 elements;
- the set of symptoms that can characterize diseases has 50 elements;
- the system has a total number of 200 diagnosers;

For each simulation we generated a matrix of cases. For consensus diagnosis and weight adjustment we use the functions previously defined in the paper. We performed the following tests:

**Accuracy of classification based on diagnoser success rate.** We measured the percentage of cases correctly diagnosed by the consensus of the diagnosers while varying the diagnosers diagnostic success rate. This showed us how successful classification varies based on the success rate of diagnosers (1).

**The result** from the graph shows that the percentage of case correctly diagnosed by consensus reaches 1 as early as the success rate is 0.3. This shows our system can reach a good rate of diagnosis even if the diagnosers have a success rate as low as 0.3 and we can see how the percentage of correct diagnosed cases increases. At a rate of 0.1 success the diagnosers will have the proficiency of a random guesser.

We generated 500 cases. We consider 100 equally distributed success rates. For each success rate, we generate the diagnosis of the expert. We assume we need 40 diagnoses to close a case. For each success rate we run 100 trials, and we average the percentage of success over the 100 trials.

**Distance between estimated and real diagnosis based on success rate of diagnosers.** The experiment measures how the distance between the real diagnosis and the diagnosis generated by diagnosers decreases. It is computed with regard to the success rate of the diagnosers. We define the distance between diagnoses by the Euclidean distance. The experiment shows how error of classification decreases when the number of cases increases. It also show the rapidity with which error rate decreases in relation to the success rate (2).

**The result** shows that except for the success rate of 0.2, all the others tend to present the same distance during the experiment. For all the success

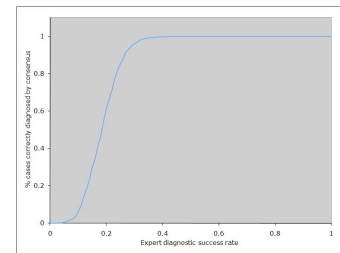


Figure 1: Accuracy of classification based on diagnoser success rate

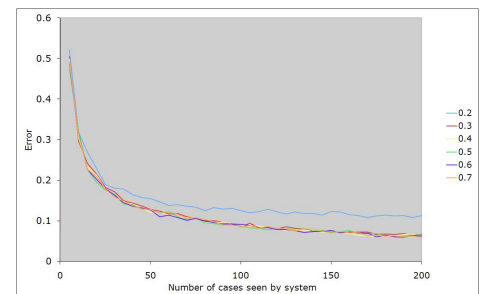


Figure 2: Distance between estimated and real diagnosis based on

rates, the curve shows a big distance between the diagnoses when the system is tested on a few cases. After testing on 50 cases the distance converges, thus the error of diagnosis is small.

We generated 200 cases. We simulated the following success rates: 0.2, 0.3, 0.4, 0.5, 0.6, 0.7. We ran 100 trials for each success rate and averaged the error over the 100 cases. We assumed we need 40 diagnoses to have a consensus on a diagnosis.

**Accuracy of decision tree based on how many diagnosers it takes to have a consensus.** This experiment simulates a situation that will probably occur often in real world usage of the system. Since the number of users diagnosing a case is variable, we are interested in seeing how the accuracy of the decision tree changes when we change the number of people that are necessary to decide a consensus, for different success rate is also different (3).

**The result** is that the accuracy of the decision tree increases with the increase in number of users that are necessary for providing the consensus for a diagnosis. This shows that in order to have an exact system we also need to have a considerable number of diagnosers voting for a case, if the users are not proficient in diagnosis. Also, fewer diagnosers need to provide a consensus if their success rate is higher. However, even if the success rate of the diagnosers is low, as low as 0.3 in our example, the accuracy of the decision tree can still be good if it had enough participants to the consensus.

The system is trained on a set of 1800 cases and tested on some other 200 cases.

**Correctly diagnosed cases based on how many diagnosers it takes to make a consensus.** This experiment tests the percentage of correctly diagnosed cases by consensus. For different values for the success rates of the users we evaluate if the consensus decided by them is the correct diagnosis. (4).

**The result** we can see is that for a rate of success of more than 0.4 the consensus is always correct if 30 people diagnose the case. However if the rate of success is smaller than 0.3, a correct diagnosis is achieved with more than 40 persons voting for the consensus.

The system is trained on a set of 1800 cases and tested on some other 200 cases.

**The general result** from the rests show that the gets accurate after it collects cases. The success rate of the diagnosers is also important but it can also be substituted up to a limit by adding more participants to the consensus. It shows we can use diagnoses from people who are not specialists because the system can actually adjust their decisions to obtain the right diagnosis. We will discuss more improvements that we plan to make in the future work.

## 5 Interactive Web Interface

The Second Opinion game serves as an interactive resource for patients seeking medical diagnostic information and as an intellectually challenging game for medical students, doctors, and anyone interested in diagnosis. In designing the game, we draw on inspiration from the 20Q design. The system was built on the CakePHP platform and we will describe its functionalities in detail.

### 5.1 Definitions

The following are terms that we will use throughout the description of the game.

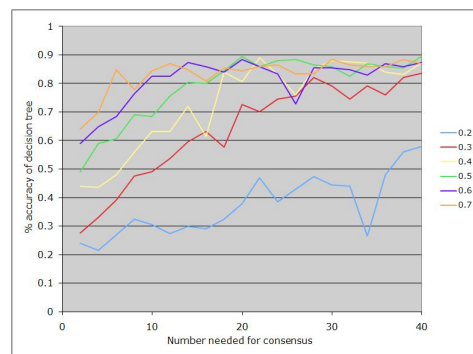


Figure 3: Accuracy of decision tree based on how many diagnosers it takes to have a consensus

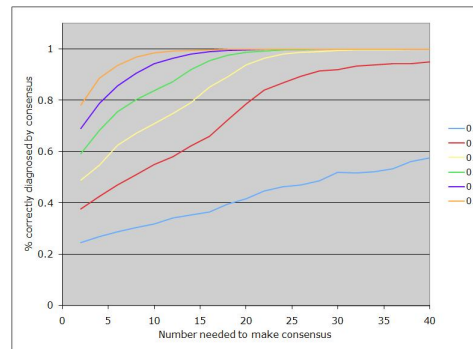


Figure 4: Correctly diagnosed cases based on how many diagnosers it takes to make a consensus

- Case** Defined by answers to a series of questions about symptoms and medical history
- Diagnosis** A hypothesis as to what disease or ailment is responsible for a given case
- Diagnosee** The user that has created the case
- Case solvers** Diagnostors that provide diagnoses for the case

We will use the ( $\diamond$ ) notation to delimit motivations on how we built the system from the rest of the descriptions of interaction flow.

## 5.2 User Interaction with Second Opinion

Second Opinion is designed to offer users the possibility to interact with the system anonymously. Second Opinion can be used from two perspectives. One is of the **diagnosee** who can provide a case and receive the diagnosis and the other is of the **case solver**, or **diagnostor**, who can diagnose cases. We have set up the system being particularly concerned for the user’s privacy. The information required from the users is minimal and most of the times optional. Thus the user is not obliged to provide any information that might identify them. The **Home Page** allows the user to either take the role of the diagnosee or of the case solver.

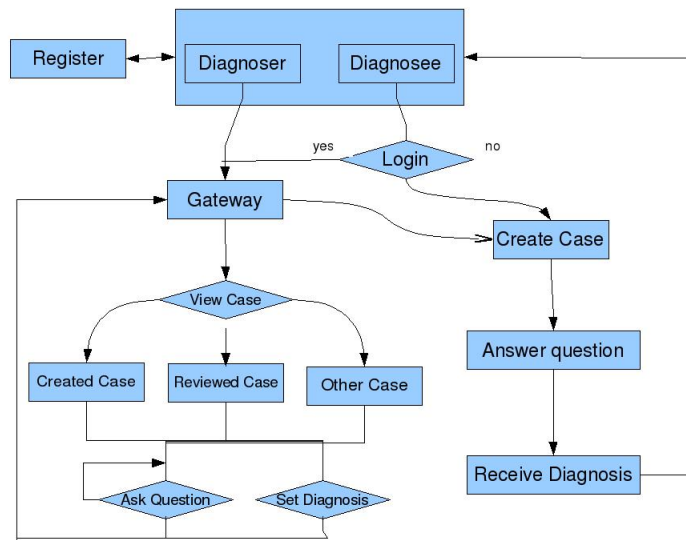


Figure 5: Flow diagram of the users interaction with Second Opinion

### 5.2.1 Diagnosee Interaction with the System

The track of the **diagnosee** consists in the following actions. The user can create a **new case**: The user is asked to provide some general information about their health condition and current medications and to agree with the disclaimer.

- $\diamond$  For now, this information is not used in the algorithm and is more orientative for the diagnostors seeing the details of the case.

After filling in the form, the user is directed to the **questions page**. The diagnosee is asked one question at a time.

- $\diamond$  The questions are aimed to allow the system to provide a better diagnosis and to allow the case solvers to have more information about the case.

The answers to questions are reduced to: *yes, no, somewhat, not sure, don't answer*. The diagnosee is free to stop answering questions at any time. The order of the questions is adjusted on the responses received from the user. The system will ask a default number of questions before providing any diagnosis.

- ◊ That is to be able to provide diagnosers with enough information such that they can provide a diagnosis. If only a very small number of questions would be answered, when the diagnoser asks questions, they cannot receive answers, especially in the cases of one time users.

The system will stop asking questions after the default number of questions or, after that, when it will be able to provide a diagnosis from the information accumulated from the user. When the diagnosee decides to stop answering questions or the system is ready to give a diagnosis, the user is directed to the **receive diagnosis page**. The system provides the diagnosis and the diagnosee can check the correctness of the diagnosis if they know it. If the provided diagnosis was wrong, the user can choose the right one from a list. If the user created the case anonymously, they are provided with a username and password to come back for a follow up of their case. Once they submit this information, the user is redirected to the home page and cannot see the details of their case unless they log in into the system.

%footnotehave to rearrange the figure

Figure 6: Question Page, Diagnosis Page, Review Page

### 5.2.2 Diagnoser Interaction with the System

The track of the **case solver** consists in the following actions. The user has to log in into the system from the main page. If the user doesn't have an account they get a user account through the **registration page** that they can access from the main page. The registration has only the username and password as mandatory requirements. The **gateway** page offers a list of cases: *cases created by the user, cases reviewed by the user and other cases*. From the gateway the user can choose to take the role of the diagnosee and create a new case, or they can view the details of any of the listed cases, which will also allow them to review open cases. By choosing to view the details of a **case created by the user**, they can see the profile of their entire case: the general information they provided, the questions they answered, with the corresponding answers, and the diagnosis, if they set one. If they haven't provided a diagnosis, they can provide one. The case solvers can assume their role by **reviewing cases**. They can either review a new case or view a case they already started to review. If they **review a new case** they will first access the **review case page**. Here they can see the general medical information of the user who created the case: medical history, medications and general symptoms. The questions answered by the diagnosee are not shown, neither is the diagnosis. If the case solvers want to review the case they will assume the role of the system, and try to guess the diagnosis by asking questions. They have the option to select questions from the list and they will receive the answer of the case creator. If the diagnosee didn't answer any of these questions, this will be stated.

- ◊ The questions are not listed directly such that they won't bias the diagnoser.
- ◊ The diagnoser should base their diagnosis on the responses to questions they find important. A more proficient diagnoser is more likely to ask the right questions and provide the right diagnosis, compared to a less proficient diagnoser that might guess based on what they would see from the questions.

When the diagnoser feels ready, they can provide a diagnosis with a certain confidence. This is aimed to help the set a better weighting of the diagnosers. Once the diagnosis is provided, it cannot be changed and the diagnoser cannot provide any other input for the case.

## 6 Other Considerations

**Patient Privacy and Transparency of Information.** We have given high consideration to the aspect of privacy. We are aware that people want to keep their information private and esitate in providing sensitive information. This is why diagnosees are not even required to register in order o receive a diagnosis. They receive a random username that they can later use itthey would like to revisit their case. During the creation of medical case, users have the possibility to provide personal data that might be helpful in diagnosis (age, sex, medications, medical history). All these are optional, along with the quesitons the system is asking. These informations can be useful in the process of providing a diagnosis, but the system can overlook them if the user does not want to provide them.

**Choosing the Medical Topic** We are using Sexually transmitted diseases(STD) for our start project. We decided to focus on this set of diseases because they are highly symptomatic and would provide a good test base for our algorithm. Also, there are several non-STD diseases that can be taken as STD and that we included. This will help test the algorithm better and we can see how sensitive it is to guessing the right disease.

We have build a set of questions that we are going to use in guessing the diagnosis. These are based on the symptoms we collected from medical books for each disease we introduced in the system. The questions are not directly associated to the symptom of a particular disease and most of them match to more than one disease. Each question will correspond to one disease in our system.

## 7 Future Work

### 7.1 Algorithm improvements and considerations

**System outputs a distribution of diseases for a new test instance.** We would like to change the algorithm such that the output is not only the most likely disease, but a distribution of diseases. This might be done by one of the following:

- Deciding on a fixed number of diagnosis  $k$  that we want to receive. This can be done by stopping the construction of the learning tree at step  $p \leq m - 1$ , when  $D^p$  has at most  $k$  elements. Assuming  $D^p = \{d_{p1}, \dots, d_{pk}\}$  we can give a distribution of these diseases by computing the likelihood of occurrence of these diseases in cases that have the same values for the symptoms  $(S_0, \dots, S_p)$  as our current instance  $(t_0, \dots, t_p)$ .

$$Likelihood(d_{pj}) = Pr(D = d_{pj} | S_0 = t_0, \dots, S_p = t_p) / Pr(D \in D^p | S_0 = t_0, \dots, S_p = t_p),$$

where  $j \in 1, \dots, k$

- Returning the set of diseases  $D^p$  when its diseases have a variance smaller than a default threshold among the cases that match our current symptoms. We could then compute the likelihood of each disease as before.
- At step  $p$ , as described in the first case, we can continue building the entire tree and compute a likelihood of the of the diseases obtained as leaf nodes.

**Create a clustered user success rate.** We would like the experts to have a precise success rate that will help us weight their diagnosis more precisely. We would like determine if experts are successful on some categories of cases and unsuccessful on other categories. Thus the success rate of an expert might be average even though for some cases the experts should have a very high success rate and for other cases a very low success rate. Secondly, some experts might be successful only on easy cases and unsuccessful on difficult cases. Thus their vote should weight less in difficult cases. Our solutions include:

- Cluster cases based on similarity  
We can consider defining an Euclidean distance between cases based on their symptoms. This way we can cluster case instances in a predefined number of clusters. Cases will be *similar* if they are in the same cluster, thus we can compare the success rate of an expert for every cluster and determine if they are more proficient for some type of cases than for others.
- Cluster cases based on difficulty of the case  
We can consider defining easy cases and difficult cases based on the voting received from experts. We can consider that cases are *difficult* if there is a high variance of diagnosis from the experts or if there are multiple

diagnosis with a high percentage of experts voting for them, and *easy* if the majority of experts agree on one diagnosis.

### 7.1.1 Other improvements

**Prioritizing cases.** We are considering giving priorities to cases by suggesting them to diagnosers. If some of the cases have not been diagnosed by any user, they will be selected and users will receive recommendations to review these cases. This way we would reduce the quantity of undiagnosed data.

Secondly we could recommend cases where the consensus diagnosis is not yet established. We could select users with a high success rate to vote in instances where only users with low success rates voted. Or, we can select what kind of diagnosers we need for a particular case and recommend the case to diagnosers having the success rate that we need.

**Undiagnosed data.** We will consider the possibility of having cases that no diagnoser provided a diagnosis for. We will look into algorithms that account for unlabeled samples and use it for learning.

**Use medical history in diagnosing.** We want to integrate the information from the medical history of the patient into the data. This could potentially help us in giving an even more precise diagnosis. We could cluster case types based on some symptoms that are present in the general information of the user or detect if some of the general information is actually correlated with particular types of cases.

**Users that have more than one disease.** One of the aspects the system might want to consider is having users with multiple diseases. This might be more likely with infectious diseases like STD's. The system might use the distribution of diseases generated from the symptoms to decide if a user has two cases. In this way it might show up that noise from the symptoms is actually useful information.

## 7.2 User Interface Considerations

**Release the online version of the game.** One of the immediate goals is to release the online game such that people can already start using it and we could start using real life data.

**Incentives for Diagnosers.** We want to find more incentives to have diagnosers often performing diagnosis. We are keeping a score for their performance. We might consider providing them with any of the following:

- statistics on how well they are doing compared to other diagnosers
- weekly statistics of their performance compared to other diagnosers
- statistics of their success rate, and success rate in particular cases
- give bonus points to users who diagnose a target number of cases per period

Also we want to find an incentive that would make the diagnosee return. In particular the ones that didn't know the real diagnosis for their case.

**Diagnoser feedback.** We can consider having feedback from the diagnosers on what they considered was the most important information from the case that helped them provide a correct diagnosis. It can come under the form of check boxes for general information, medications, general symptoms, questions they answered. This can be optional for diagnosers. If there is a database of such information it can be used to compare what the decision tree algorithm finds important in the symptoms, or we can prioritize questions also using this information.

**Advertise the site.** We want to have as many users as possible to use the game. We will consider advertising the site on medical forums and to students studying medical sciences.

**Allow users to provide the diagnosis after a case has been closed.** Having the real diagnosis for case is valuable, since it represents a sure source of correct diagnosis. We are normally not allowing any modifications to a case once it has been closed. This is because once the case is closed all the weights of the diagnosers are readjusted. If the real diagnosis is provided and is actually different from the consensus one, we would have to find a way of readjusting the weights of the diagnosers. This might be difficult because they already influenced the system through the current case and their success rate.

**Allow new questions.** We will consider accepting new questions from the diagnoser to the user. This would require processing of the questions. It would also give the first step towards having interaction between users. We might think about extending this to live interaction, which would bring up several new perspectives on the system.

### 7.3 Simulation improvements

**Use a distribution of rate of success.** We are assuming that all the diagnosers have the same rate of success. We want to test the system on data that has a uniform distribution. We will test the system's behavior when the diagnosers have a different rates of success.

**Use data that doesn't fit the distribution of the diseases and symptoms.** The diseases' symptoms respect the distributions that we generate initially. We want to test the system on diseases that give a new distribution of symptoms than before. We would like to be able to test the system when the data doesn't have enough information to be classified or when it is very noisy.

**Use public medical statistics on diseases.** We want to compare the data that we obtain from our system to the data that is available in medical journals or books regarding the distribution of diseases and of symptoms to diseases.

## References

- [1] EW Nawar, RW Niska, and J Xu. National Hospital Ambulatory Medical Care Survey: 2005 Emergency Department Summary. *Advance Data from Vital and Health Statistics*, (386), 2007.
- [2] K Schrock. Twenty Questions, Ten Million Synapses. <http://scienceline.org/2006/07/28/tech-schrock-20q/> (accessed on 1-April-2008).
- [3] L. von Ahn and L. Dabbish. Labeling images with a computer game. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, 2004.
- [4] L. von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64, 2006.
- [5] B.G. Buchanan and E.H. Shortliffe. *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley series in artificial intelligence)*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1984.
- [6] Jia Wenyan, E. ScLabassi, T. Kanal, and Ozkurt M.L. and Scheuer Mingui Sun. An intelligent user interface system for diagnosis of epilepsy. *Bioengineering Conference, 2006. Proceedings of the IEEE 32nd Annual Northeast*, 2006.
- [7] Stephen O. Agyei-Mensah and Frank C. Lin. Application of Neural Networks in Medical Diagnosis: The Case of Sexually-Transmitted Diseases. *Australian Physical and Engineering Science in Medicine*, 15(4), 1992.
- [8] Wojtek Michalowski, Steven Rubin, Roman Slowinski, and Szymon Wilk. Triage of Acute Abdominal Pain in Childhood: Clinical Use of a Palm Handheld in a Pediatric Emergency Department. In *HICSS*, 2004.
- [9] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in Medicine*, 23:89–109, 2001.

- [10] Steven L. Salzberg. Book Review: C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. *Machine Learning*, 16(3):235–240, 1994.
- [11] M. Li and Z.H. Zhou. Improve Computer-Aided Diagnosis with Machine Learning Techniques Using Undiagnosed Samples. *IEEE Transactions on Systems, Man and Cybernetics*, 37(6), 2007.
- [12] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.