CPSC 470 • December 2017

A System to Diagnose Patients with Cough

Stylianos Rousoglou

Yale University
stylianos.rousoglou@yale.edu

Advisor: Professor Dragomir Radev

Abstract

This paper explores approaches to designing and implementing a computer system that allows patients with cough to diagnose themselves without input from a medical expert. After performing literature review and studying tangential research efforts, we explore approaches to modelling the diagnosis process. This involves patient, symptom, and diagnosis representation, as well as building a prototype that encodes and utilizes relevant expertise to provide reasonable diagnoses to the user. To tackle the complex task of diagnosing an arbitrary patient, we simplify the problem to patients with a given "principal symptom", specifically cough. We first attempt to build a simple classification model using machine learning techniques. However, this proves to be an extremely challenging task, given the lack of sufficient, real-life, well-formatted data, and the absence of a reliable mechanism for us to collect such data. Even learning paradigms that require minimal training data fail to produce any meaningful results, given the large number of parameters of even the simplified version of the problem. As a result, we transition to a probabilistic model, and encode domain knowledge in the form of medical expertise (approximating statistical evidence). Our prototype overcomes the challenge of lacking realistic data by relying on statistical evidence that is widely collected and easily retrievable. In addition, it greatly facilitates extensibility and refinement of the model, to allow for further experimentation and potential future use. Though this work focuses on patients with cough, the approaches discussed, software implementation, and conclusions drawn can be generalized to apply to other principal symptoms, and thus to generic patients.

I. Introduction

Doctors Pusztai and Howard of the *Yale School of Medicine* approached us in August 2017 with the long-term objective of developing a patient diagnosing system. The ultimate goal of their project is to replicate a physician's approach to questioning a patient and offering the most accurate diagnosis available, using a computer model. By encoding the relevant medical experience and expertise, the model will enable patients to diagnose themselves by entering relevant information about their background, symptoms, and triggers, and will finally arrive at one or more possible diagnoses, each associated with a likelihood probability. The desired system has to account for at least as many parameters as an actual physician does; patient history and previous medical records; family medical history records; current, as well as past, patient symptoms, their progression and potential recurrence; patient demographic information (age, occupation, etc.); environmental triggers; and chronic illnesses/conditions that may help illuminate present symptoms.

In order to simplify the problem and focus our efforts on accuracy, rather than breadth of our prototype, we simplify the original definition by considering a special class of patients, specifically those whose "principal symptom" is cough. The use of the principal symptom, which the doctors defined as the "main symptom of the patient", as a starting point for our application, was inspired by the fact that doctors use the principal symptom themselves to determine the sequence of questioning to be followed during diagnosis. By focusing our research on popular real-world algorithms for diagnosing cough, and collecting data and findings specific to our subset of patients, we render the problem significantly more manageable. At the same time, by presenting our findings in a general fashion, and designing our web application with a doctor-facing interface that allows for additions, modifications, and refinements, we ensure that the software can be extended or adapted to include an arbitrary number of additional principal symptoms, questions and diagnoses.

The main challenge faced in this project was the lack of patient data, enough to build a machine learning classifier, specific enough for the purposes of parametrizing and training our model, or accurate enough to guarantee good results. We initially experiment with a number of classification approaches, including a Decision Tree Classifier and a Bayesian Network prototype, using data produced by us and manually labeled by the doctors. However, such learning models that generally work well even with small data sets, proved to be of little use, given the large number of parameters our model has to account for.

We thereby transition to a probabilistic model that allows us to use statistical data collected from any reliable source, or alternatively doctors' experience from studying or treating particular diseases. Specifically, we associate answer-diagnosis pairs with an estimated relative likelihood of the symptom/answer and the disease being observed simultaneously in a given patient. This way, not only can we rely on decades of data collection and expertise, but we can also refine the model indefinitely, as discussed in sections III, IV, and V, to achieve better results.

II. Related Work

In performing literature review, we learned about different approaches to building a classifier, as well as examined tangential research conducted in other areas of medicine that could provide insights and inform our own approach.

We started by exploring Bayesian Network models as they have been used in Triage systems in the past [Sadeghi et al., 2006] that displayed promising results. In "*A Bayesian model for triage decision support",* data from clinic charts was used to build a Bayesian Network classifier, which then attempted to Triage (i.e. prioritize ER visitors based on the severity of their medical condition) new patients. Though the dataset used had several limitations, including its small size and the need for extracting information from documents that are frequently ambiguous, the model generally performed comparably well to the human nurse in classifying patients. Characteristics of this approach that are particularly appealing are its ability to tolerate a small data set, as well its flexibility in incorporating new data into the model, which facilitates refinement and extensibility.

We also read about Decision Tree models and studied their advantages and limitations, as well as relevant applications in the context of multiclass classification. Song et. al., 2015 offer a practical, in-depth look into the techniques and best practices of building classifiers using a Decision Tree, and discuss ways the particular approach is suited for applications in medical research. Meanwhile, "*Decision trees in epidemiological research"* discusses a particularly relevant application of DT. Venkatasubramaniam et al. (2017) attempt to build a model that partitions a population into groups with similar values of some outcome variable. The problem shares several characteristics with our own; for instance, approximating non-linear relationships, or estimating the effects of both continuous variables (e.g. age) and categorical data (e.g. gender) is a challenge shared by the two works. After contrasting several techniques and evaluating the models' effectiveness, the paper concludes that although the trees' results could be helpful, they were not to be thought of as conclusive or consistently reliable.

"A Learning Health Care System Using Computer-Aided Diagnosis" discusses the relevant topic of computer-aided diagnosing, and provides several useful insights into the development of a system that can improve upon a physician's accuracy in reaching a conclusive diagnosis. After outlining several limitations of existing systems, Cahan and Cimino (2017) introduce a theoretical framework for developing diagnosis support systems in the future, and present schematics of proposed entity relationship models for such systems. A particularly interesting insight is offered by the chart-like depiction of the conceptual framework proposed (Diagram 1); the right half of the model describes likelihood estimations, and is therefore closely related to our ultimate approach and the associated prototype application we developed.

There is an increasing number of complicated medical tasks that intelligent systems can carry out accurately. Very recently, the Machine Learning group at Stanford developed CheXNet, which excels at specific medical tasks; for instance, it performed better than medical experts in identifying pneumonia using chest X-Rays. However, the CheXNet algorithm was trained using a large dataset of chest X-Rays accompanied by real ground-truth labels, the actual diagnoses of the patients. On top of that, the level of specificity of the particular problem definition is also helpful in limiting the parameters involved in the proposed solution; a model to diagnose patients generally has to account for countless more features. Expectedly, not many commercial software tools claim to be capable of accurately diagnosing generic patients (i.e. have broad enough capabilities to lead to any diagnosis); that is the ultimate goal of Pusztai and Howard.

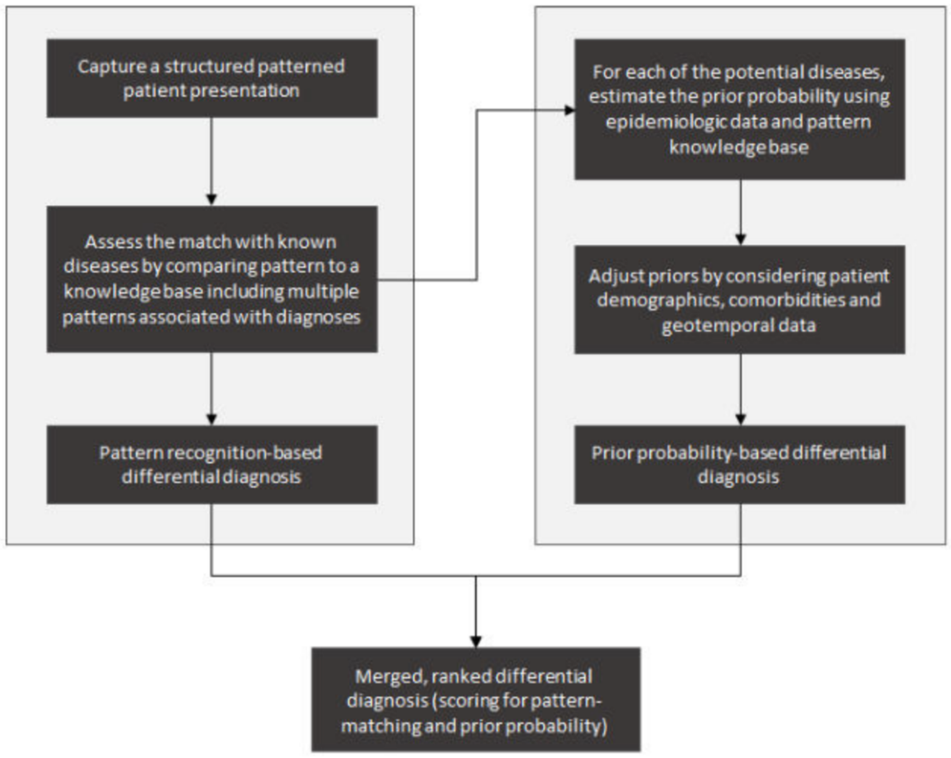

Capture a structured patterned patient presentation
Assess the match with known diseases by comparing pattern to a knowledge base including multiple patterns associated with diagnoses
Pattern recognition-based differential diagnosis
For each of the potential diseases, estimate the prior probability using epidemiologic data and pattern knowledge base
Adjust priors by considering patient demographics, comorbidities and geotemporal data
Prior probability-based differential diagnosis
Merged, ranked differential diagnosis (scoring for pattern-matching and prior probability)

Diagram 1: Conceptual framework for building computer-aided diagnosis systems. The rightmost half of the diagram makes uses of prior probability distributions and patient information to estimate the likelihood of encountering a disease in the given patient

As part of our preliminary research, we searched for and trialed relevant commercial software applications available on the internet. *iTriage* is an iOS application that offers a wide variety of medical information, including searching for hospitals and doctors, symptoms, diseases, and medication. The user-interface is simple to use and provides many additional capabilities, such as tracking one’s medical history and booking appointments. We also considered *Doctor AI,* a recurrent neural network based self-diagnosing project that utilizes real medical data and patient records collected over 8 years to diagnose patients. Though this approach is closer to what we hoped to ultimately pursue, the lack of a primary medical data source prevented us from attempting to train a complex RNN model, as similar training data could not be obtained.

III. Approach

A. Representation Models

<u>10 – question</u>

The first step to building a prototype was to establish a model for representing patients, diseases, and symptoms, as well as all other relevant information that would traditionally be provided by a patient. We begin by modelling a simple Triage interaction of a patient with a doctor or nurse. We always assume that the patient's principal symptom[1] is *cough.* We first ask the patient for some general information, presented in *Figure 1*.

We then proceed to asking about symptoms associated with cough, similar to how a nurse would in the context of Triage. For our simple model, we use 10 questions, shown in *Figure 2*, and seek a Yes/No answer from the patient. We then attempt to estimate the urgency of their present condition based on the observed symptoms and background data provided. We used a simple Google Form to create around 20 test patients to be used for training. Pusztai and Howard then labelled each patient case with a triage score on a scale of 1 (no need for treatment) to 5 (ER emergency), providing our ground truth value.

Name	Age	Gender	Body Temperature	Regular Smoker?

Figure 1: General information in Triage scenario

Cough?	Headache?	Chills?	Sore Throat?	Vomitting?	Muscle Pain?	Tiredness?	Runny Nose?	Heartburn?	Chronic cough?

Figure 2: Symptoms related to cough in Triage scenario

We quickly refined the model to add several dimensions to the patients' responses. After consulting with the doctors, it was clear that more specific information on a patient's

[1] Primary reason for seeking medical assistance

condition would be necessary to appropriately triage them. To that end, we added three dimensions to each Yes/No answer; severity, trajectory, and continuity. For each of the symptoms shown in *Figure 2,* three additional questions were asked, shown in *Figure 3.* These are standard follow-up questions that should be asked for all symptoms; according to the doctors, they provide information that's valuable to evaluating a patient's condition. Responses to the first two questions need not in reality be categorical; however, for the purposes of our model, we allowed a choice between three options: "Mild", "Medium", and "Intense", and "Better", "Same", and "Worse", respectively.

How intense?	Are these symptoms getting..?	Chronic?

Figure 3: Additional model dimensions in Triage scenario

<u>100 – question</u>

We moved on to adopting a more detailed model, developed with the help of Pusztai and Howard, which uses 97 responses as model features for a given patient (dubbed the 100 – question model). The nature of these features is not limited to symptom information, but also includes questions related to background, family history, and social habits. Answers are once again meant to be binary, but increasing the feature space now enables us to pose highly specific questions that include the dimensions presented in *Figure 3*. A selected subset of these questions is shown in *Figure 4.*

Female?
Exposure to allergen?
Exposure to irritant?
Occurs in the morning?
Worse when lying down?
Worse after exercise?
Recent abdominal surgery?
Non-smoker?
<20 pack years?
20 - 40 pack years?
Asbestos exposure?

Figure 4: Features in diagnosis scenario

Similarly, we increase the precision of our predictions; we abandon the 0 – 5 scale and use a list of diseases as categorical outcomes of the predicted output variable, the diagnosis. Recall, we are attempting to diagnose a patient, rather than to simply triage them. *Figure 5* displays some of the 23 diagnoses associated with the principal symptom of cough in our final model.

Viral infection
Post-viral cough
Influenza
Pertussis
Pneumonia
TB
Bacterial sinusitis
Stable asthma
Asthma exacerbation
Stable COPD
COPD exacerbation

Figure 5: Diagnoses in diagnosis scenario

B. Decision Tree Classifier

First, I implemented a Decision Tree classifier to triage patients from the 10 – question model presented above. A tree was among the most appropriate machine learning approaches available given our lack of sufficient data to train more complex models like neural networks. Decision trees can be built with arbitrarily few data points, and have an intuitive visual representation; they can also be easily refined or reconstructed.

After importing the model patient responses (which we created) into Python, some data manipulation was required before building the classifier. Incomplete data had to be identified and replaced with neutral values (mostly "No"s). Textual responses were converted into categorical values (specifically integers) to be compatible with the classifier. Finally, ground truth labels were imported and associated with each patient. After several trials using tiny datasets we created, a typical decision tree produced by our approach looked something like the tree depicted in the *Figure 6*. The *sklearn* Python module was used for training; *numpy* and *pandas* were used for data manipulation.

The main limitation of this approach is that the model produced, though 100% accurate on training data, may not learn to make decisions about important parameters. Specifically, if the training data of the Decision Tree Classifier is not diverse, then the model will not learn how to use features that are scarce in the training population, and therefore will perform poorly on diverse testing datasets. Visualizing the DT below helps identify a significant barrier; the vast parameter space of our model, in conjunction with the lack of extensive and appropriately detailed training data, render any decision tree we produce of little use. As we can see, the training data points are first split by body temperature, then by intensity of cough; finally, some subset of the tree is further split by age. Other parameters, including most all experienced symptoms, have not been learned on; in other

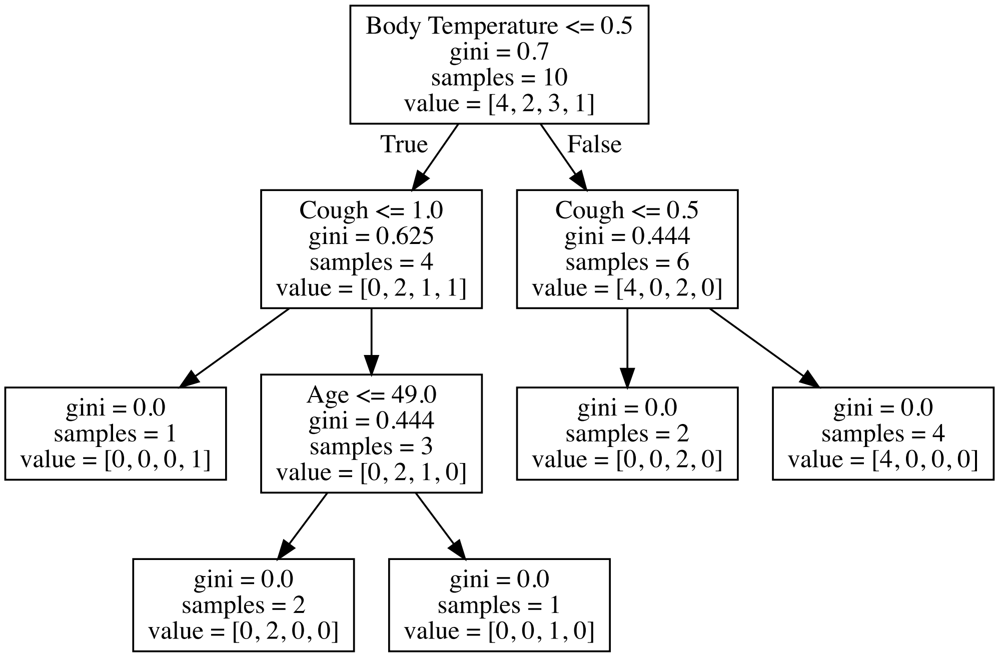
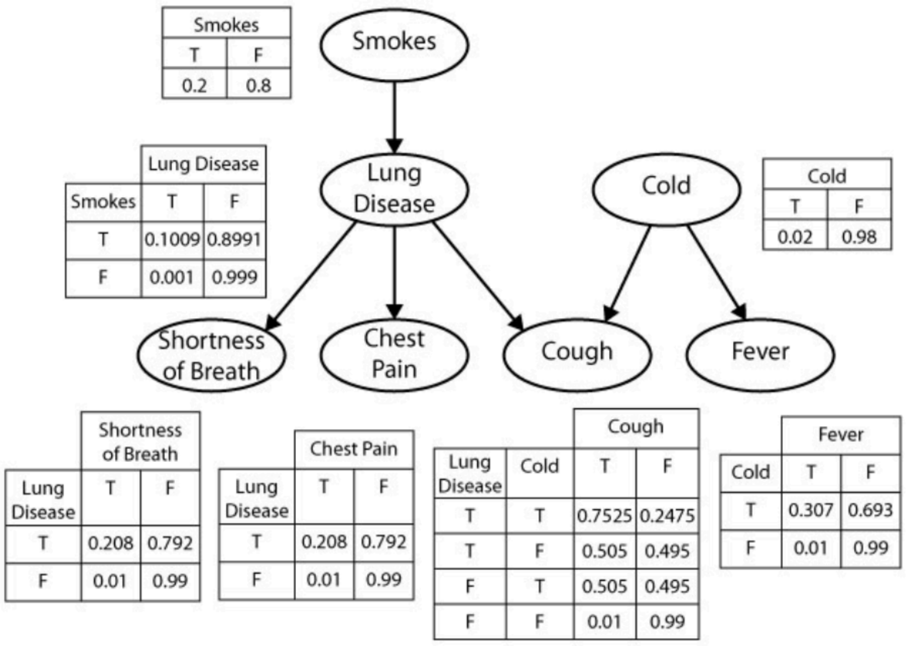


Figure 6: A visualization of the learned decision tree model on our tiny dataset

words, our tree can classify all our made-up patients accurately without even considering their symptoms. That is clearly a failure to build a model sensitive to all relevant parameters, mostly stemming from the lack of adequate data discussed above.

C. Bayesian Network



Smokes	
T	F
0.2	0.8

	Lung Disease	
Smokes	T	F
T	0.1009	0.8991
F	0.001	0.999

Cold	
T	F
0.02	0.98

	Shortness of Breath	
Lung Disease	T	F
T	0.208	0.792
F	0.01	0.99

	Chest Pain	
Lung Disease	T	F
T	0.208	0.792
F	0.01	0.99

		Cough	
Lung Disease	Cold	T	F
T	T	0.7525	0.2475
T	F	0.505	0.495
F	T	0.505	0.495
F	F	0.01	0.99

	Fever	
Cold	T	F
T	0.307	0.693
F	0.01	0.99

Figure 7: Simple Bayesian Network likelihood estimator

My teammates, Allen Wang and Irene Li, explored other popular techniques for building a classification system, most notably the Bayesian Network model for classification. In the Bayesian Network, patient data is used to compute conditional probabilities that relate different parameters of the model with each other, as well as with the predicted output variable. Evidence about the state of the system (i.e. the patient's symptoms, background, family history, social habits etc.) is then used to make predictions. Specifically, statistical tools such as likelihood maximization are used to produce a probability distribution of the possible values of the output variable, in our case the diagnosis. A visualization of a simple Bayesian network demo performed on made-up data from out 10 – question model is shown in Figure 7. Expectedly, the accuracy of the model is sensitive to the quality and quantity of data; once again, the lack of a sufficiently large dataset of real patient information prevented us from pursuing this solution further.

D. Maximum Likelihood Estimator

To overcome the difficulties encountered in training Decision Tree and Bayesian Network classifiers, we had to consider approaches that did not require a large dataset of recorded patient symptoms. As a result, we moved away from machine learning solutions, and considered a probabilistic model that remedies the situation, allowing us to produce a much more detailed and meaningful diagnosing prototype.

For the purposes of building a likelihood estimator, we developed and used the 100 – question model outlined earlier. Each principal symptom p_i is associated with a vector of questions $\boldsymbol{q}$ and a vector of diagnoses $\boldsymbol{d}$. Moreover, each tuple (p_i, q_j, d_k) is assigned a likelihood value $l_{p_i,q_j,d_k} \in (0, \infty)$, the likelihood that a patient with principal symptom $\mathrm{p_i}$ and an affirmative answer to $\mathrm{q_j}$ is diagnosed with $\mathrm{d_k}$, *relative to the entire pool of patients with* p_i. To compute the relative likelihood of each $\mathrm{d_k}$ for a given patient, we first produce vectors $\boldsymbol{r_{q,d_k}}$ of likelihoods applicable to the patient, based on their responses to each question q_j; specifically, $\mathrm{r}_{q_j,d_k} = l_{p,q_j,d_k}$ if the patient answered "Yes" to $\mathrm{q_j}$; otherwise, $\mathrm{r}_{q_j,d_k} = 1$.

$$\mathrm{r}_{q_j,d_k} = (Answer\ to\ q_j\ is\ \text{"Yes"}\ ?\ l_{p,q_j,d_k}\ :\ 1)$$

We then define the likelihood of some diagnosis d_α for the given patient as follows:

$$lik(\mathrm{d}_\alpha) = \frac{1}{\sum_k \prod_j \mathrm{r}_{q_j,d_k}} \prod_j \mathrm{r}_{q_j\,d_\alpha}$$

In other words, given a relative likelihood table l_p and a list of patient answers (evidence observed), we consider the product of the applicable likelihoods for each diagnosis, and normalize by the sum of such likelihood estimations for all diagnoses. We defined the result to be the *relative likelihood of the given patient to suffer from d_k, given his principal symptom p_i and his responses to each question q_j.*

The rationale behind building a likelihood estimator was that statistical information about the demographics, symptoms, background and social history of populations diagnosed with the same condition, is generally available by a variety of medical sources. Moreover, such statistics have been recorded for years, so their validity can be cross - checked and evaluated. There is little ambiguity in interpreting the data; for instance, "how much more likely than a random individual is it for a smoker to be diagnosed with lung cancer?" is a clear question with a single numerical answer. In addition, such statistical information can be approximated by a doctor's expertise. Therefore, in the absence of recorded data, or for testing model variability, a physician can use their knowledge and experience to estimate such relative likelihood values. In fact, that is the case with the "cough" likelihoods table; the likelihood values encoding the relationship between questions and diagnoses in our final model were produced by Howard. Of course, meticulous research performed by a physician could gather more accurate likelihood estimations. Therefore, given enough time and sources, the model can be refined and improved indefinitely.

The limitations of this model are clear. The approach requires us to make several assumptions; first, we assume that listing all possible diagnoses associated with a principal symptom is feasible, which is not realistically the case. The same is applicable to questions a patient is asked; even the 100 – question model doesn't come close to exhausting the potentially relevant questions a real-life patient could be asked by a physician, and that's only for the simplified problem of diagnosing cough. In addition, the model fails to consider connections between answers to different questions, i.e. conditional likelihoods that may affect the resulting calculation significantly. For instance, the answer to "how much more likely than a random individual is it for a smoker to be diagnosed with lung cancer?" could differ significantly based on the gender of the examined patient; such insights are not captured by the current approach, and could only be incorporated by adding features (questions) with higher specificity to the model, along with their conditional likelihoods.

Another limitation of such a probabilistic solution is that is doesn't encode any qualitative expertise that is useful to efficiently arriving at an accurate diagnosis. As we learned by studying exiting algorithms for triaging patients, different answers to specific questions often affect the sequence of questioning a doctor would perform, triggering relevant follow-ups and or necessary clarifications. Such patterns can't be learned by a probabilistic model, which implies that the questioning process will likely be slower and less efficient, occasionally posing unrelated questions, or failing to ask conditionally relevant ones. Though the described qualitative knowledge can be approximated by manually designing a conditional question sequence on the patient-facing side of the system, attempting to do so would be tedious and likely really hard, as the possible scenarios increase exponentially with the number of questions asked and answers allowed.

IV. Web Application

The prototype application we produced was designed to be simple, extensible, and robust. It follows a traditional 3-tier architecture, outlined in *Diagram 2* below.

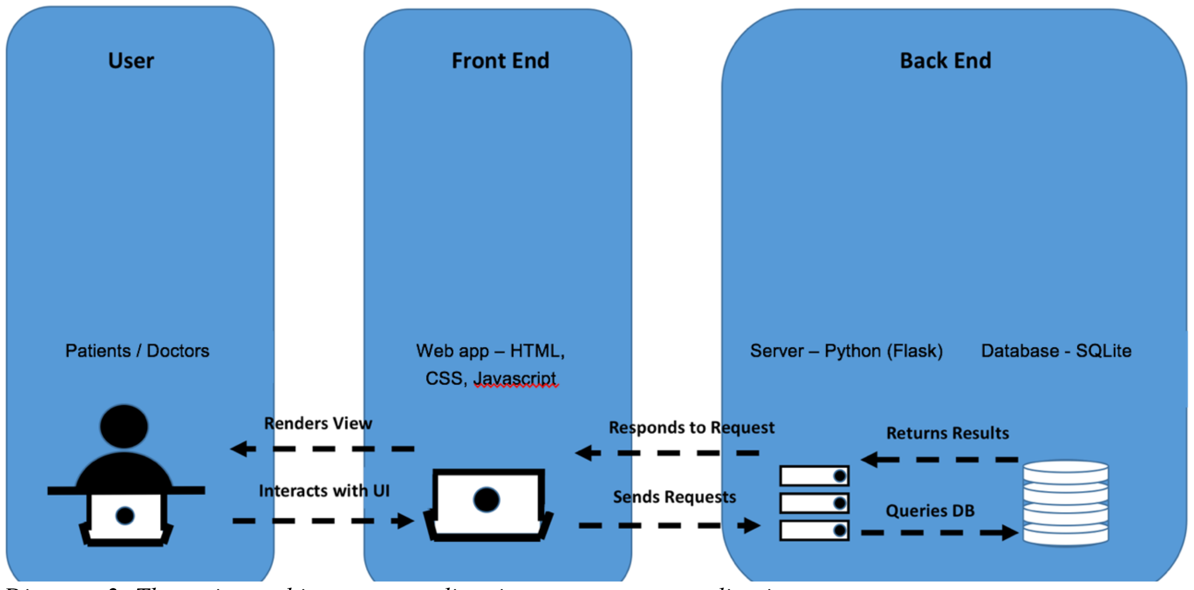


Diagram 2: Three-tier architecture paradigm in our prototype application

The application is based on the probabilistic model discussed in III(D), and was designed to be lightweight, user-friendly, and easily extensible. The following table outlines the requirements of the prototype application, as they were discussed and revisited through meetings with Pusztai and Howard, and our individual contributions to development:

ID	Description	Priority	State	Contributors
1	Web application easily accessible over the web	Must	Complete	Allen, Stelios
2	Entity representation adapted for efficient database storage and retrieval (using SQLite)	Must	Complete	Allen, Stelios
3	Robust server functionality that performs calculations and returns results efficiently	Must	Complete	Stelios
4	Client-facing service for patient self-diagnosing; user-friendly interface, comprehensive questions	Must	Complete	Allen
5	Display comprehensive diagnosis results to patient, along with corresponding calculated probabilities	Must	Complete	Stelios
6	Ability to register and log users in with personal credentials	Should	Complete	Stelios
7	Ability to remember previous diagnoses of returning users; history available to display	Should	Complete	Allen
8	Doctor-facing interface for refining or adding to the existing model	Should	Complete	Allen
9	Modular code design to facilitate future maintenance and extensibility	Should	Complete	Allen, Stelios
10	Application deployment	Must	Complete	Allen, Stelios

Database

We store all data in a local database file named *triage.db.* We use SQLite to avoid the added complexity of relying on a SQL server, and to ensure the application is versatile, easily deployable and environment-independent. I devised a data storage scheme that allows for maximal flexibility in designing a database API and manipulating data, while eliminating data repetition. Specifically, tables *principals, diagnoses,* and *questions,* hold the textual values of all principal symptoms, potential diagnoses, and diagnostic questions respectively, and map them to a unique integer value for easy reference and manipulation. The schema for the three tables is shown in Figure 8. The *likelihoods* table then maps likelihood tuples l_{p_i,q_j,d_k} (see Section III) to likelihood values in a simple schema shown in Figure 9. This design minimizes data duplication, and allows for efficient data retrieval.

Field	**Type**	**Not NULL**	**Primary Key**
id	INTEGER	Yes	Yes
name	TEXT	Yes	No

Figure 8: Structure of principals, diagnoses, and questions tables

Field	**Type**	**Not NULL**	**Primary Key**
principal	INTEGER	Yes	Yes
diagnosis	INTEGER	Yes	Yes
question	INTEGER	Yes	Yes
likelihood	REAL	Yes	No

Figure 9: Structure of tuple-to-likelihood mapping in likelihoods table

To allow users to maintain an account on our application, we created the table *users,* to store a (unique) username, hashed password, and quality (patient/doctor) for each registered user (see Figure 10). We also maintain the history of diagnoses performed by each user using 3 tables, *history, patient_input,* and *patient_results*; this design (see Figure 11) ensures we can efficiently store an arbitrary amount of provided answers and produced diagnoses for each patient, once again minimizing data repetition.

Field	**Type**	**Not NULL**	**Primary Key**
id	INTEGER	Yes	Yes
username	TEXT	Yes	No
hash	TEXT	Yes	No
doctor	BOOLEAN	Yes	No

Figure 10: Structure of users table

The design of the database tables is an important element of the prototype application. With the proposed schemas, it becomes easy to add new data (including users, principals, diagnoses, questions, and likelihoods) as well as to represent relationships between them (e.g. add new $l_{p_i,q_j,d_k} \rightarrow v \in (0, \infty)$ mappings). As such, the model is not

Field	Type	Not NULL	Primary Key
id	INTEGER	Yes	Yes
user_id	INTEGER	No	No
principal_id	INTEGER	No	No
history	INTEGER	No	No
question	INTEGER	No	No
response	INTEGER	No	No
history	INTEGER	No	No
diagnosis_id	INTEGER	No	No
diagnosis_name	TEXT	No	No
probability	REAL	No	No

Figure 11: Structure of history, patient_input, and patient_results tables respectively

limited to the data it currently contains. It can be refined indefinitely by improving likelihood mappings, adding new (and more specific) questions, enriching the list of diagnoses, etc.

Back-end (server)

My work focused on the database schema (detailed above) and the design and development of the server functionality. Python was chosen for our server because of the wide variety of data manipulation, server framework, and database libraries available in the particular language. We used *Flask*,[2] a Python server micro-framework, as our server's particularly lightweight backbone.

For communication with the database, I developed a simple SQLite API for retrieving and modifying data in the SQL tables (found on the bottom of *application.py).* We broke down the server functionality into simple endpoints, and separated model logic from server functionality by maintaining distinct "method" and "endpoint" sections in the code. The all-

[2] http://flask.pocoo.org/

around modularity of the code offers significant flexibility in factoring-out and reusing parts of it, while ensuring that potential future enhancements are straightforward to perform.

Front-end (client)

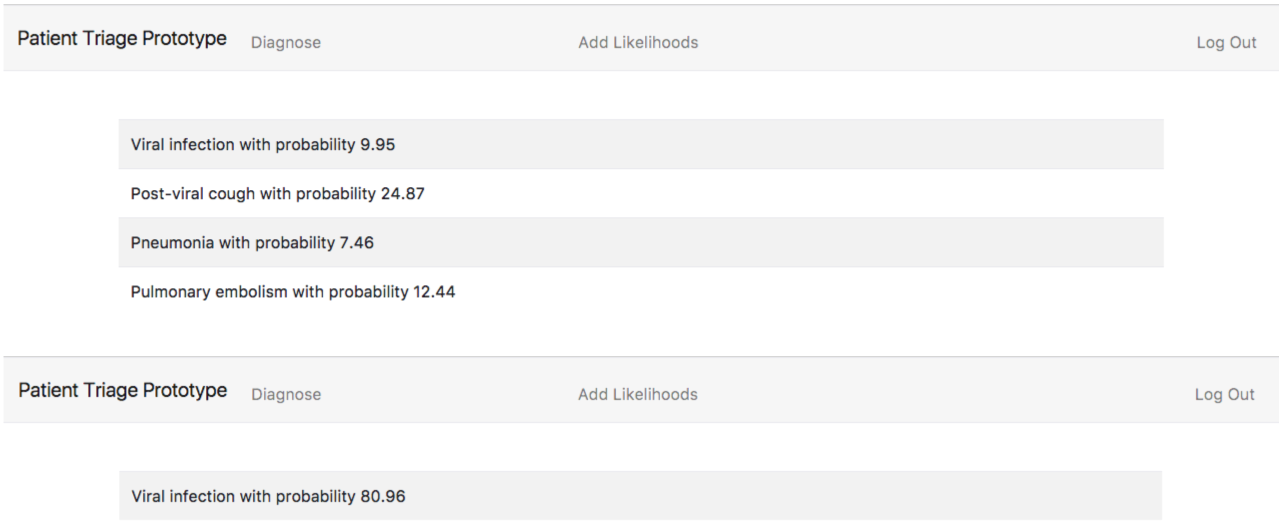
The client-facing part of the prototype (front-end) was developed by my partner, Allen Wang. HTML and CSS, as well as the templating language *Jinja* were used for webpage rendering.[3] The client's two main components are the patient-facing and doctor-facing interfaces. The "Diagnose" interface can be used by both types of users to self-diagnose and produce likelihood estimations for the available diseases; in addition, doctor accounts can refine the stored data model by editing existing likelihood mappings, as well as adding options to the principals, diagnoses, and questions tables. Data validation is also performed in the front-end using *Javascript*, ensuring fault-tolerance while eliminating the need for unnecessary communication with the server.

Screenshots of the final product are presented on the next page. *Figure 12* depicts the results page (where the probability distribution over potential diagnoses is displayed) after the patient has responded to all questions and the likelihood calculations have been performed. The history feature of our prototype is displayed in *Figure 13,* while one aspect of the doctor-facing interface that allows for modifications of the existing model (in this case, editing likelihood mappings) is shown in *Figure 14.*

Plan for deployment

We plan to deploy the application in December 2017, likely on *Heroku* or a similar cloud platform. For development and testing, a workspace on the Cloud9 IDE was used (*c9.io*).

[3] http://jinja.pocoo.org/


Patient Triage Prototype Diagnose Add Likelihoods Log Out

Viral infection with probability 9.95

Post-viral cough with probability 24.87

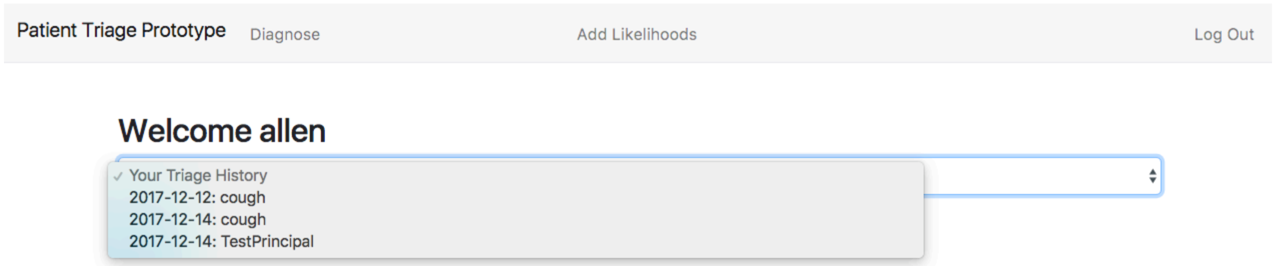
Pneumonia with probability 7.46

Pulmonary embolism with probability 12.44

Patient Triage Prototype Diagnose Add Likelihoods Log Out

Viral infection with probability 80.96

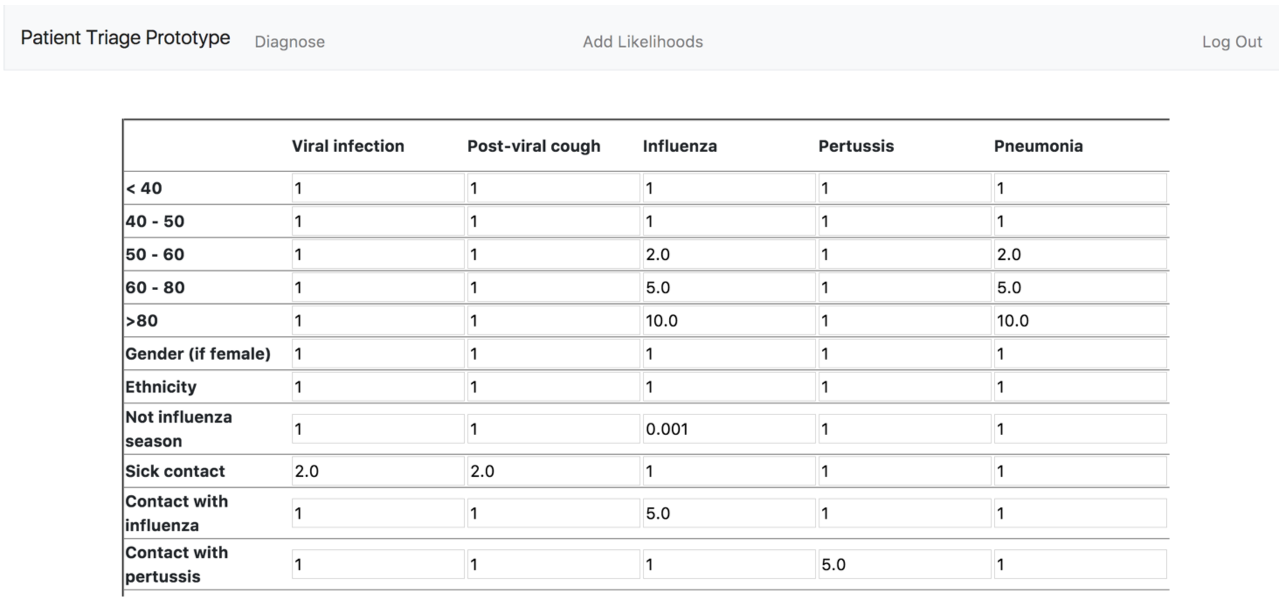
Figure 12: Examples of inconclusive (top) and conclusive (bottom) diagnostic results produced by the prototype application, in the form of percentage likelihoods of particular diagnoses


Patient Triage Prototype Diagnose Add Likelihoods Log Out

Welcome allen

✓ Your Triage History
2017-12-12: cough
2017-12-14: cough
2017-12-14: TestPrincipal

Figure 13: Screenshot of user history feature, whereby a registered user can revisit past results


Patient Triage Prototype Diagnose Add Likelihoods Log Out

	Viral infection	Post-viral cough	Influenza	Pertussis	Pneumonia
< 40	1	1	1	1	1
40 - 50	1	1	1	1	1
50 - 60	1	1	2.0	1	2.0
60 - 80	1	1	5.0	1	5.0
>80	1	1	10.0	1	10.0
Gender (if female)	1	1	1	1	1
Ethnicity	1	1	1	1	1
Not influenza season	1	1	0.001	1	1
Sick contact	2.0	2.0	1	1	1
Contact with influenza	1	1	5.0	1	1
Contact with pertussis	1	1	1	5.0	1

Figure 14: Screenshot of the doctor-facing interface for modifying likelihood values in the model

V. Future Work & Conclusions

Lacking relevant testing data and objective truth labels to use in evaluating our prototype on, we are required to perform manual tests and consider the produced results individually. In 10 trial runs, the application returned reasonable results, according to our test subjects and the doctors; on one occasion, it successfully diagnosed Professor Radev's viral infection (with an estimated probability > 80%), while in another, it produced multiple results (a probability distribution) that Pusztai and Howard deemed as "very reasonable suggestions".

The most important outcomes of this project are the conclusions we can draw about the nature of the problem at hand, and the suggestions we can make regarding pursuing the next stages of this long-term effort. Indeed, the different techniques we experimented with, though often unsuccessful, led us to define the problem much more precisely; identify many of the potentially infinite parameters involved; acquire a much better understanding of the requirements of a feasible solution; and take the first step toward the ultimate goal of building a complicated model for general self-diagnosis.

The main limitations of the current approach are detailed in section III(D). However, improvements to the model can also be proposed. Via meticulous collection of evidence, a more accurate likelihood estimator can be built. If rigorous statistical results eventually replace the doctors' rough approximations of likelihood values, the model will likely provide better estimates of the posterior probability of each diagnoses for a given patient. We can also envision the model accounting for conditional probabilities, combining existing questions with relevant information such as place of origin, genetics, and other factors that may alter likelihood values significantly, to potentially improve results. With that in mind, a user-friendly interface was created to facilitate modifications to the model, while eliminating the need for the doctors to ever interact with the database or the code directly. By adding questions with higher specificity, the precision of the likelihood estimator can be improved.

In planning next steps and moving the project forward, we strongly advised Pusztai and Howard to pursue access to real-life patient and diagnostic data. A good dataset will allow for the development of a more complex model, presumably based on some Recurrent Neural Network architecture. Two sources the doctors are currently considering are handwritten reports from patient examinations, and electronically submitted forms of background and symptom information from patient visits to a hospital. The former approach poses an interesting NLP challenge, and has the advantage of virtually unlimited availability of training data. The latter approach is more convenient in that the data is electronically stored, and therefore more easily parsed; in addition, data points and information collected will be much more standardized in the case of electronic forms than in that of doctors' individual notes. In either case, models based on real patient diagnoses will not only be able to extract patterns that may even be foreign to doctors, but will also automatically learn to simulate doctor – patient interactions more effectively. Such models may also leverage continuity, making use of previous diagnoses (and related statistical observations) to better estimate likelihoods.

VI. Acknowledgements

This project was worked on by myself, Stylianos Rousoglou, in conjunction with Allen Wang (Yale '18) and Irene Li (Yale CS PhD Candidate), under the supervision of Professor Dragomir Radev. I would like to thank Allen and Irene for a great semester, as well as Professor Radev for his encouragement, guidance, and continued support. Special thanks to Lajos Pusztai and Fred Howard of the Yale School of Medicine for sharing with us their vision and medical expertise, and devoting time and energy to this project.

VII. References

"About Itriage - Itriagehq". 2017. *Itriagehq*. http://about.itriagehealth.com/.

Cahan, Amos, and James J Cimino. (2017). "A Learning Health Care System Using Computer-Aided Diagnosis". *Journal Of Medical Internet Research* 19 (3): e54. doi:10.2196/jmir.6663.

Choi, E., Bahadori, M.T., Schuetz, A., Stewart, W.F. & Sun, J.. (2016). Doctor AI: Predicting Clinical Events via Recurrent Neural Networks. Proceedings of the 1st Machine Learning for Healthcare Conference, in PMLR 56:301-318

Langarizadeh M, Moghbeli F. Applying Naive Bayesian Networks to Disease Prediction: a Systematic Review. *Acta Informatica Medica*. 2016; 24(5):364-369. doi:10.5455/aim.2016.24.364-369.

Rajpurkar, Pranav, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, and Daisy Ding et al. 2017. "Chexnet: Radiologist-Level Pneumonia Detection On Chest X-Rays With Deep Learning". doi:1711.05225v2.

Sadeghi Sarmad, Afsaneh Barzi, Navid Sadeghi, and Brent King. 2006. "A Bayesian Model For Triage Decision Support". *International Journal Of Medical Informatics* 75 (5): 403-411. doi:10.1016/j.ijmedinf.2005.07.028.

Seixas, Flávio Luiz, Bianca Zadrozny, Jerson Laks, Aura Conci, and Débora Christina Muchaluat Saade. 2014. "A Bayesian Network Decision Model For Supporting The Diagnosis Of Dementia, Alzheimer′S Disease And Mild Cognitive Impairment". *Computers In Biology And Medicine* 51: 140-158. doi:10.1016/j.compbiomed.2014.04.010.

Song, Yan-yan, and Ying Lu. “Decision Tree Methods: Applications for Classification and Prediction.” *Shanghai Archives of Psychiatry* 27.2 (2015): 130–135. *PMC*. doi:10.11919/j.issn.1002-0829.215044.

Venkatasubramaniam, Ashwini, Julian Wolfson, Nathan Mitchell, Timothy Barnes, Meghan JaKa, and Simone French. 2017. "Decision Trees In Epidemiological Research". *Emerging Themes In Epidemiology* 14 (1). doi:10.1186/s12982-017-0064-4.