A System to Diagnose Patients with Cough

Allen Wang and Stylianos Rousoglou
Yale University, New Haven, CT
allen.wang@yale.edu
stylianos.rousoglou@yale.edu

Dragomir Radev (Advisor)
Yale University, New Haven, CT
dragomir.radev@yale.edu

1. Abstract

Despite the increasing growth of technological capabilities in the modern world, most of today’s patients will still go through the same diagnosis experience as patients from many years ago. Patients stop by their hospital of choice and a medical professional usually asks them a series of questions. Doctors will then combine the data gathered from the patient and their medical expertise to arrive at a conclusion, usually a set of probable diagnoses or recommendations for further examination. The expertise of medical professionals is essentially a black-box system that takes in the status of a patient and produces results based on years of medical training and first-hand clinical experience. The motivation of this project was exploring the possibility of reducing the margin of human error in doctor diagnoses. Such a system would combine various machine-learning and natural language processing (NLP) techniques with existing patient diagnosis data to exceed the performance of human doctors while minimizing the risk of human-error.

We sought to investigate the feasibility of creating a system that can be incorporated into hospital settings for patient diagnosis. Our initial proposal was focused on the emergency-room context, a setting where the accuracy and speed of diagnoses are arguably most important. A hypothetical emergency-room triaging system would triage patients with priority based on condition severity and resource necessity [Emergency Severity Index, 2012].

Our project eventually moved towards creating a system that provides diagnoses based on a principal symptom. We reasoned that to eventually succeed in a time-sensitive emergency-room environment, a hypothetical diagnosis system should first be able to accurately diagnose a patient in a normal hospital check-in situation.

Our work focused on diagnosing a patient with a cough, one of the most common principal symptoms. This paper primarily covers our various approaches to creating this cough diagnosis system, the challenges we encountered, our proof-of-concept web-application, and discussions on related and future work in this domain. Allen’s contributions to these respective domains are covered in this paper while Stylianos’ specific contributions are detailed in an analogous paper.
2. Introduction

This paper primarily covers Allen’s work on the various pieces of our project. After extensive research on related works, machine-learning approaches, and iterative discussions with Lajos Pusztai and Frederick Howard, our partners at the Yale School of Medicine, we created a functional web-application that produces a set of diagnoses for cough as the principal symptom. Throughout this process, we explored decision trees, Bayesian networks, and maximum likelihood estimation as well as the results of related works. Irene Li was our final team member and worked with me on investigating Bayesian techniques.

Our web-application is extensible for future work to incorporate more symptoms. The Github repository for the project can be found here: https://github.com/wallen/triage

Due to the lack of patient data, we were personally unable to implement machine learning techniques to our system, but uncovered the additional requirements necessary to train and improve our system. The contributions of Li, Pusztai and Howard are acknowledged.

3. Related Works

I evaluated several related projects and research papers in the domain of clinical decision-making assistance technology. This section summarizes my conclusions drawn from each work and details insights that influenced the project.

3.1 Free-Text for Emergency Sepsis Support

In Creating an automated trigger for sepsis clinical decision support at emergency department triage using machine learning [Horng et al., 2017], doctors experimented with using free-text in emergency department triaging. Their specific scenario pertained to identifying patients with potential sepsis infections (bacteria and harmful toxins in infected wounds). Free-text is structureless data that is often readily available but difficult to draw conclusions from. In a hospital setting, clinician notes are the primary source of free-text. After preprocessing the notes, these doctors used the bag-of-words and topic models to represent the data. They measured the area under a support vector machine receiver operating characteristic (ROC) curve to calculate the discriminatory power of their models. Besides learning their conclusion that using free-text yielded the best performance models, I discovered requirements that pertained to our objective, which was emergency-room triaging at the time. Besides the NLP techniques needed, this experiment had access to data from over 230,000 patient visits. I realized that it was extremely difficult outside of a hospital environment to obtain this large of a data set to train models. Specifically, since we could not conduct a structured experiment to specify what data to record from each patient, we hoped to utilize existing patient diagnosis data as sources of truth to train our models.
3.2 RNN for Predictive Diagnosis and Medication

Similarly, Doctor AI [Choi et al., 2016] introduced a different machine learning technique while also reinforcing the necessity of a reliable data source. Doctor AI is a predictive temporal model based on recurrent neural-networks (RNN). Like the sepsis clinical decision support experiment, Doctor AI utilized a large data-set: specifically, time stamped electronic health-record (EHR) data from over 260,000 patients and 2800 physicians to train the RNN. Doctor AI predicts a set of medications and diagnoses given a patient’s individual medication and diagnosis history. Unlike the previous paper, the data sourced from EHR is more structured. We hypothesize that a future system could combine both free-text processing and RNN training to achieve greater performance than that of systems implementing only one of the individual techniques.

3.2 Bayesian Network for Triaging

Bayesian Network for MSK Triage, a presentation on using Bayesian networks for musculoskeletal triaging [Marsh and Joseph, 2013] inspired us to explore using a Bayesian Network in our system. As we realized the limited timeline of our semester project meant we would not be able to access large datasets, we turned to a Bayesian Network approach. This powerpoint demonstrated to us that combining a deterministic diagram of various patient conditions and their conditional probabilities allows a predictive model to be generated without the need of large amounts of training data.

All of these related works influenced our iterative progress throughout the course of this project. We each explored different methods and ultimately refined our system while understanding future requirements for improvements.

4. Machine-Learning Approaches

There are many machine-learning techniques for creating predictive models. For this project, our scope was limited by the lack of patient data available. Our partners Pusztai and Howard could not grant us access to real patient data from the Yale School of Medicine as we were not affiliated with the medical school ourselves and because the Health Insurance Portability and Accountability Act of 1996 (HIPAA) protects patient medical information. Despite this hurdle, we were able to experiment with two machine-learning techniques: Bayesian networks and decision trees.

4.1 Bayesian Approaches

In exploring the use of Bayesian networks, we were able to gather some relevant data from Howard. Specific to cough, Figure 1 lists the likelihood for a diagnosis if a corresponding question or condition is true. The base likelihood ratios are relative to 1, meaning a value of 0.5 means the diagnosis is half as likely if a given question/condition is true.
These likelihoods were intended to be a starting point for building a Bayesian network. Based on one of the related works that implemented a Bayesian network for musculoskeletal assessment, I sought to build a Bayesian network for our cough-based principal symptom diagnosis. The 92 questions gathered from doctors represented variables that contributed to 1 of 23 possible diagnoses. This information was iteratively refined and reviewed by Howard, Pusztai, and myself.

Li and I then encountered a problem when visualizing the Bayesian Network. Since we had 92 variables that all contributed to the diagnoses, we needed the conditional probabilities of each diagnosis given the possible values of these 92 variables. Even simplifying these variables into discrete “yes” or “no” values and neglecting continuous variables, each diagnosis would enumerate $2^{92}$ rows. After I created a potential conditional probability table for a single diagnosis (Figure 2), we realized that Bayesian networks could be implemented with either training data to learn from or a regimented set of conditional probabilities; unfortunately, we had no feasible way of obtaining either for the duration of this project. An alternative solution, discussed in the following section on Probabilistic Approaches and proposed by Li, ended up also lacking the sufficient conditional probabilities.
4.2 Decision Tree Approach

As for decision trees, Stylianos created a decision tree in Python to triage cough patients into 5 classes based on severity. I added code from the Python graphviz library to his source code to generate the decision-tree in the figure below.
This model was created after we realized our lack of access to patient data, and trained from sample data. The sample data was parsed from an Excel spreadsheet linked to a Google Form that we populated with questions. These questions were gathered from our discussions with doctors from the school of medicine based on their own experience diagnosing patients.

There was some over-training on our decision tree as the root decision splits the samples based on age. In reality, age is not the most deterministic factor when diagnosing a cough, but due to our limited sample data size, the decision tree’s recursive partitioning property chose the attribute that splits the data set most evenly. In our case, age was the attribute that provided the most information gain when split (partitions the data most evenly).

5. Probabilistic Approaches

5.1 Maximum Likelihood Estimator

After realizing our limitations in creating a Bayesian network that required conditional probabilities for all 92 questions of possible patient input, we instead assessed the feasibility of implementing a Bayesian maximum likelihood estimator (MLE). My proposed maximum likelihood estimator, shown below, had a slightly different structure than the previous Bayesian network.
Figure 4: Conditional Probability for a Question Given 23 Diagnoses - Slightly Less Unfeasible

The rationale behind a maximum likelihood estimator is that we would evaluate the likelihood of diagnoses given a patient's history, social data, as well as observed symptoms from questions. After processing a patient's input to the symptoms, a SAT solver on each answered question could produce the set of diagnoses for each question which could then be compared to find the most likely diagnoses. In the MLE diagram (Figure 5), the diagnoses would be our parameter space, and the data space would be the patient's answers for the 92 questions.

Figure 5: Diagram of a Maximum Likelihood Estimator
An example Bayesian maximum likelihood estimator and the conditional probabilities for a few select variables are shown in Figures 6 and 7. Essentially, instead of having to account for $2^{92}$ conditional probabilities for 23 diagnoses, a maximum likelihood estimator would require $2^{23}$ conditional probabilities for 92 questions. Despite this improvement, we still were unable to acquire the conditional probabilities or learn them without training data.

![Figure 6: Bayesian Maximum Likelihood Estimate Diagram](image1)

![Figure 7: Conditional Probabilities for Figure 6](image2)
5.2 Relative Likelihood Predictor

Without adequate training data required for most machine-learning techniques, we adopted a predictive algorithm using the provided relative likelihoods. A patient’s input for the questions produces a vector containing ninety-two 1’s and 0’s based on discrete “yes” and “no” answers. We then multiplied this vector by the ninety-two question relative likelihoods for each diagnosis. Values that corresponded to a 0 would then be replaced with a 1 since that 1 meant the likelihood did not increase nor decrease. Each diagnosis is then summed up and normalized to get a percentage. The diagnoses that met our cutoff percentage (5% for our application) were then filtered as a result set. This method ended up being the predictive model for our application and the diagnosis probabilities which can be conclusive (Figure 8) and not as conclusive (Figure 9).

![Figure 8: Viral Infection Diagnosis Results](image1)

![Figure 9: Multiple Diagnoses Results](image2)

This approach does have its issues. Each question is independent of all other questions as conditional probability is not accounted for. The relative likelihoods themselves were also generated by a single doctor and may reflect his own biases. However, given our lack of readily available training data, this relative likelihood predictor gave us an accurate predictive model to build our web-application around.

6. Our Work

Our web-application implements the relative likelihood predictive model for cough diagnosis. A patient user can register and log in to an existing account. On the homepage, the patient can select a prior
diagnosis from a dropdown and see the results of that diagnosis. Patients can also start a new diagnosis by clicking the ‘Diagnose’ tab and going through the questions.

I also added support for a doctor interface. Along with all of the patient functionality, a doctor can click the ‘Add Likelihood’ tab and either update or add a new set of principal symptom relative likelihoods. Our application requirements and individual responsibilities are shown in the following MoSCoW table.

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

Figure 10: MoSCoW Style Requirements Table

6.1 System Architecture

We developed the front end with JavaScript, HTML, and CSS. Stylianos set up a Python server using Flask which interacted with an SQLite database. Our code was stored in the Github repository noted in the Abstract section, and we tested it locally using cloud9’s integrated development environment.
6.2 Front End

I implemented the user-interface and front-end code almost exclusively. All of the views were implemented in HTML with Flask’s Jinja template engine. After initial prototype feedback from Pusztai and Howard, I added a doctor interface for updating and adding likelihoods, as well as the history of each patient’s diagnosis results. For user-friendliness, I grouped the questions into sequentially served pages to recreate the Google Form experience. This required storing previous answers as Jinja template arguments as the user navigated the questions.
Figure 13: Add/Update Likelihood Selection

<table>
<thead>
<tr>
<th></th>
<th>Viral infection</th>
<th>Post-viral cough</th>
<th>Influenza</th>
<th>Pertussis</th>
<th>Pneumonia</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 40</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>40 - 50</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>50 - 60</td>
<td>1</td>
<td>1</td>
<td>2.0</td>
<td>1</td>
<td>2.0</td>
</tr>
<tr>
<td>60 - 80</td>
<td>1</td>
<td>1</td>
<td>5.0</td>
<td>1</td>
<td>5.0</td>
</tr>
<tr>
<td>&gt; 80</td>
<td>1</td>
<td>1</td>
<td>10.0</td>
<td>1</td>
<td>10.0</td>
</tr>
<tr>
<td>Gender (if female)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ethnicity</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Not influenza season</td>
<td>1</td>
<td>1</td>
<td>0.001</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sick contact</td>
<td>2.0</td>
<td>2.0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Contact with influenza</td>
<td>1</td>
<td>1</td>
<td>5.0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Contact with pertussis</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5.0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 14: Update Existing Likelihoods
6.3 Back End

In the Python server, I added specific route handlers to interact with the front end, especially for various grouped question pages. To implement user diagnosis history, I also had to modify the SQLite database. I created a patient_input table and a patient_results table that contained a foreign key history_id. This history_id key linked to a history table which stored the principal symptom and user for this particular diagnosis. This extra history table relates users to all of their diagnoses and results. Based on which prior diagnosis is selected on the front end, the corresponding results are then queried.
Figure 17: ‘patient_input’ Table Structure in Database

Figure 18: ‘patient_results’ Table Structure in Database

6.4 Overall Project Work Delegation

Stylianos and I both attended each meeting with our advisor and partners and were always at similar levels of understanding our project progress and future work. This allowed us to split up work as evenly as possible.

Stylianos and I discussed and shared responsibility for the following project decisions:
- Machine-learning methods to explore
- Shifting to probabilistic models after discovering the lack of available data
- Choosing a web-application platform for the deliverable
- System design and development stack

His work primarily focused on:
- Implementing the decision tree model
- Converting the relative-likelihoods into code to generate diagnosis probability calculations
- Setting up the server and application environment
- Designing the Entity-relationship diagram

My own work was centered around:
- Researching related works to shape the direction of the project
- Assessing the Bayesian Network and Maximum likelihood estimator models
- Implementing the patient-interface for our final web-application
- Adding the doctor-interface for updating and adding likelihoods

7. Conclusion & Future Work

7.1 Conclusion

Our research into the different machine-learning and probabilistic approaches provided not only a basis for our web-application, but also knowledge for future development and improvements. Understanding the deficiencies of our system and the factors that limited alternative solutions provide targets for future improvements centered primarily on finding reliable data-sources.

The final web-application not only serves as a system for diagnosing patients with cough, but also is a valuable proof-of-concept to demonstrate the potential of clinical-decision support systems. And if this potential leads to future funding, our proof-of-concept will only be a small part of the value produced from this project.

7.2 Future Work

The immediate next steps will be to migrate our system from our local environments and host it on Tangra. Besides that there is a lot of room for future development and integrating machine-learning techniques provided that reliable sources of data are acquired.

One possibility that we have discussed, drawing inspiration from the sepsis decision support project [Horng et al., 2017], is exploring NLP methods to process free-text clinician notes. This would open up a range of models that we did not get a chance to implement. Paired with the capability of adding and updating likelihoods, machine-learning techniques could help refine the relative likelihoods of the existing predictive model by assigning likelihoods based on actual results and trained experience. Getting a reliable source of data would multiple different predictive models, besides Bayesian networks, decision trees, and maximum likelihood estimators, to exist on the system. Conclusions and diagnoses could be drawn from multiple models and if parameterized and weighted properly, could allow for accurate, comprehensive and timely patient diagnoses.

Finally, expanding beyond a cough to other principal symptoms would make the system more practical for hospital usage. After enough time and iteration, it could even be introduced into an emergency-room context to triage patients. But for now, there is a lot of work between our current system and a widely-established clinical decision support system.

8. References

**Doctor AI: Predicting Clinical Events via Recurrent Neural Networks.** [online] PMLR. Available at: http://proceedings.mlr.press/v56/Choi16.html

