

What's Up, Doc? A Medical Diagnosis Bot

Monica Agrawal
agrawalm

Janette Cheng
jancheng

Caelin Tran
cktt13

Abstract

Medical chatbots, conversational agents built with medical applications in mind, have the potential to reduce healthcare costs and improve accessibility to medical knowledge. We built a text-to-text diagnosis bot that engages patients in conversation about their medical issues and provides a personalized diagnosis based on their symptoms and profile. Our chatbot system was able to identify symptoms from user inputs with an average recall and precision of 64% and 66%, respectively. Using these extracted symptoms, correct symptom codes were identified with a recall of 64% and a precision of 72%. Finally, the chatbot returned the expected diagnosis 34% of the time. This demonstrates that a medical chatbot can somewhat accurately diagnose patients with simple symptom analysis and a conversational approach without the need for radio buttons and other graphical tools, which suggests that a effective spoken language medical bot could be viable. Moreover, the relative effectiveness of this bot indicates that more advanced automated medical products may grow to serve a bigger role in healthcare.

1 Introduction

In the United States, healthcare is a highly advanced and rapidly developing field, but Americans face limited and expensive coverage. In 2016, healthcare expenditure exceeded \$10,000 per capita. (Keehan et al., 2016) In addition, 4.5% of Americans failed to obtain necessary medical treatments due to high costs. (CDC, 2015) Due to the need for lower costs and easier access to care,

the healthcare space holds a lot of opportunity for effective and convenient automated systems.

We built a text-to-text conversational agent that diagnoses patients explaining their condition using natural language. The bot asks for relevant information, e.g., age and sex, and requests a list of symptoms. The system remembers past responses and asks progressively more specific questions in order to obtain a good diagnosis. The three primary components of our system are (1) identification and extraction of symptoms from the conversation with the user, (2) accurate mapping of extracted (and potentially ambiguous) symptoms to documented symptoms and their corresponding codes in our database, and (3) developing a personalized diagnosis as well as referring the patient to an appropriate specialist if necessary.

Although several medical diagnosis chatbots already exist, including Your.MD, Babylon, and Melody, current implementations focus on quickly diagnosing patients by identifying symptoms using tools such as radio buttons and pure system initiative questions instead of natural conversation. (Gallego, 2016) (Furness, 2016) Our system focuses solely on the analysis of natural language to extract symptoms, which could make it easier for elderly, less technical users to communicate their symptoms as well as make it relatively straightforward to support spoken language by adding ASR and NLG components. In its current form, our bot's best application would be as a preliminary diagnosis tool that patients could use to assess their symptoms before going to the doctor, perhaps using the bot's specialist referral feature to choose the right care provider.

2 Background

There are a number of different paradigms for building dialog systems. According to Allen et

al., one of the most developed and robust is the frame-based approach, which includes most of the spoken systems constructed to date. The system essentially interacts with the user just enough to acquire sufficient information to perform a specific action or set of actions. A simpler dialogue paradigm is the finite state system, where states determine machine responses and handwritten logic interprets user input to choose the correct transition to the next state. In order to maximize our control of the logic with which our system identifies symptoms, our approach uses a finite state paradigm for managing the doctor-patient dialogue.

In order to properly identify and extract desired information from the user, we must have a good natural language understanding approach that accurately interprets information tokens and intents. As a review, intents are global properties of user inputs that signify the goal(s) of the user. (Bhargava et al., 2011)

When communicating symptoms, patients may communicate several at the same time and not all may be in the same category, i.e., patients may have multiple intents. For example, patients may try to convey the bodily location of a symptom as well as the description of the symptom itself. Xu et al handles multi-intent classification by exploiting shared intents across different intent combinations, which differs from the naive strategy of treating combinations as individual labels. Our system uses a catch-all approach that compares user input to a database of potential symptoms, actively screening for symptoms from various categories and sidestepping this problem of multiple intent classification. (Xu and Sarikaya, 2013)

As for interaction design, Yankelovich et al. identify a set of challenges and principles for designing speech user interfaces (Yankelovich et al., 1995). These observations were collected in the evaluation of SpeechActs – a speech-only application for email, calendar, weather, and stock quotes. Although our chatbot uses text input and output, many of the researchers’ suggestions for speech dialogue apply to the general design of conversational agents.

One of the main challenges is simulating conversation, which requires maintaining “shared context.” In terms of dealing with errors in speech recognition/understanding, users preferred “progressive assistance” over repeated “I didn’t

understand” responses. The example progression give in the paper is the following:

“What did you say?”

“Sorry?”

“Sorry. Please rephrase.”

“I didn’t understand. Speak clearly, please.”

This sort of feedback gave users the sense that the system was trying to understand what they were saying, avoiding the “brick wall” effect. We used elements of progressive assistance in our bot’s user feedback templates, especially in the symptom gathering phase.

Another recommendation is “verification commensurate with the cost of the action which would be effected by the recognized utterance.” In other words, the more serious the consequences (e.g., deleting data or categorizing the user as the wrong sex), the more explicit the verification. Explicitly verifying every single input is annoying from a user experience perspective. As an alternative, we echo back parts of commands and inputs in our answers to demonstrate the system’s understanding of what the user has said or requested.

3 Chatbot Design Overview

Our chatbot’s user dialogue is a linear design (see Figure 1) that proceeds from symptom extraction, where we use Wit.ai and a logic layer to extract symptom-like entities, to symptom mapping, where we identify corresponding symptom codes in our database, to diagnosis, where we use ApiMedic, (Priaid, 2016) an external diagnosis engine, to identify the most likely diagnosis based on the user’s age, sex, and symptoms.

3.1 State Design

We used a finite state model (see Figure 2) for our chatbot’s dialogue design. In order to efficiently move the conversation towards an accurate diagnosis, we handcrafted all the logic for state transitions, templates for natural language generation, and system initiative to prompt easily interpretable responses from the user.

Besides its greetings and goodbye states, our agent has three main conversational phases: acquisition of basic information, symptom extraction, and diagnosis. Our bot starts off by asking about

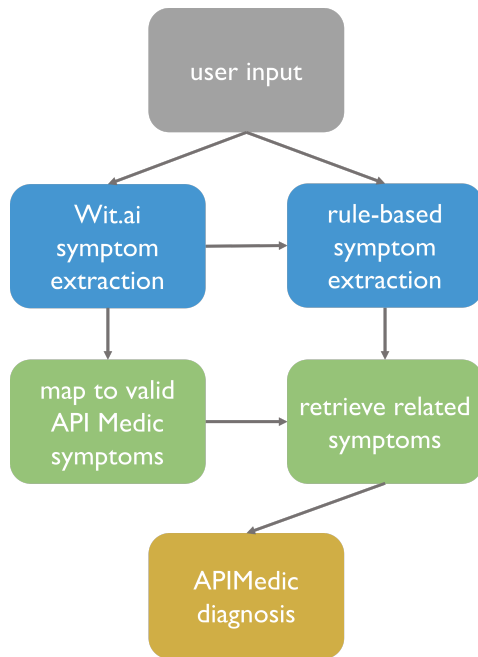


Figure 1: Chatbot program flow

the user’s age and sex and then enters a loop of symptom extraction states until it acquires sufficient information for a diagnosis. Users have the option of entering the loop again to talk to the doctor about another set of symptoms after receiving their first diagnosis.

3.2 Demographic Extraction

Slot filling for demographic information, age and gender, is conducted with relatively naive algorithms due to the simplicity of the tasks. There is generally a set number of terms one uses to describe biological sex, and age is generally only described numerically or with the words spelled out. We chose not to include implicit and explicit confirmation for these steps, since users generally felt confident in bot’s ability to accurately understand their inputs. To do so would disrupt the flow of conversation.

3.3 Symptom Extraction

Symptom extraction refers to the mechanisms used to identify potential symptom substrings in users’ natural language text input. In cases where users simply list their symptoms, perhaps after some leading phrase (e.g. “I have a cough, fever, and nausea”), this is a relatively simple task. However, the system should also be able to handle input like, “When I read, I’m okay at first, but over time, my eyes seem to get tired, and I start to see

double.” In this case, the system should extract substrings like “eyes tired” and “see double” (and not substrings like “read” or “okay”).

In order to handle a variety of symptom-description styles – especially more naturalistic input – we use a combination of rule-based extraction and Wit.ai entity extraction. Our rules extract a symptom if all the words of a valid ApiMedic symptom occur in user input. The Wit.ai component is trained on hand-generated and labeled symptom descriptions and uses both substring matching and free text search to identify potential symptom entities. The details of substring and entity extraction are further discussed in Section 4 on our iterative development process.

3.4 Symptom Mapping

ApiMedic accepts 270 different symptoms, which range from general descriptions, such as “muscle pain”, to very specific conditions, such as “bleeding in the conjunctiva of the eye”. As a result of misspellings and differences in phrasing or word choice, the symptoms users describe do not always map to valid ApiMedic symptoms that can be fed into the diagnosis engine. For example, if a user says, “I’m feeling woozy”, the extracted symptom substring would be “woozy”, which is not a valid input to ApiMedic. Thus, in order to diagnose the user, extracted symptom substrings must be mapped to valid ApiMedic symptoms.

Given some extracted substring from the user’s input, we generate a list of suggested closest symptoms using edit distance and cosine similarity of GloVe word vectors. We then ask the user to confirm if they have any of the suggested symptoms. We also perform an amount of mapping during symptom extraction using WordNet synsets, simple stemming, and permutations of multi-word symptoms to generate synonyms (Wit.ai can map multiple synonyms to the same keyword). The details of each of these strategies are further discussed in Section 4.

In addition to mapping user-described symptoms to their ApiMedic counterparts, we also utilize user-described symptoms to suggest other related symptoms (retrieved through ApiMedic) that patients may have.

3.5 Diagnosis

Since the models for diagnosis were outside the scope of this project, we relied on an outside API

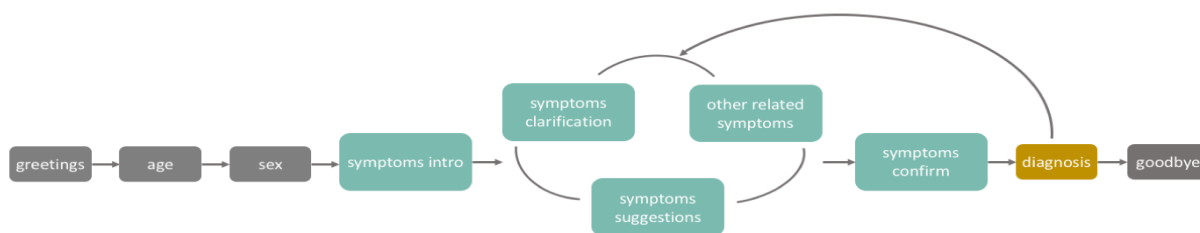


Figure 2: Finite state graph

available through ApiMedic that takes in demographic information and symptom lists as inputs and returns likely diagnoses. The platform provides 420 different diagnoses, and our chatbot outputs the top 1 or 2 most likely diagnoses. In some scenarios, there are unusual combinations of symptoms that yield no diagnosis. As part of the diagnosis step, if there is a specific specialist that would be helpful, e.g., an ophthalmologist or a gynecologist, we suggest the patient request a follow-up appointment with them.

4 Iterative Process

One of the most challenging issues we faced was creating a system that could process the wide variety of ways users might describe symptoms. Thus, the main tasks we iteratively improved upon were symptom extraction and mapping.

4.1 Wit.ai

We first used Wit.ai (trained on hand-generated and labeled examples of symptom descriptions) to extract symptom entities from user input. Specifically, we were using the “keyword” and “free text” search strategies, a combination of checking for matches to symptoms in a pre-defined list and extracting substrings not part of this list. While using keywords allowed us to extract exact matches to symptoms from ApiMedic and Wikipedia, it could not capture different ways of expressing the same symptom. Adding “free-text” helped with this issue, but also resulted in the extraction of words that are not symptoms. For example, from the sentence, “I’ve been having trouble with diarrhea and indigestion,” Wit extracts “trouble”, “diarrhea”, and “indigestion”. Because Wit tends to extract more false positives than false negatives, it is used as a catch-all for substrings that could potentially be symptoms.

4.2 Handwritten Rules

Even though all valid symptoms are included as keywords in Wit, these symptoms are at times only extracted if they appear exactly in user input. This can be an issue for multi-word symptoms, such as “limited ankle mobility”. To handle cases like this as well as single-word symptoms, the very first extractor we run goes through the list of ApiMedic symptoms and checks to see if all of the words for a given symptom occur in the input while ignoring their relative positions. For example, the ApiMedic symptom “limited ankle mobility” would be extracted from the user input, “my ankle has limited mobility”. We then remove words corresponding to the extracted symptom(s) and pass the remaining input to Wit to extract any remaining substrings this step may have missed, e.g., original input: “my ankle has limited mobility” → input sent to Wit: “my has”.

4.3 Edit Distance

We wanted to be able to take into consideration potential misspellings by users (e.g., dizzyness for dizziness) and allow there to be flexibility in the part of speech used to describe a symptom (e.g., nausea vs nauseous). Therefore, when mapping extracted symptoms, assuming there were no perfect matches, the edit distance is found between possible symptom candidates and the extracted symptom. If the edit distance is 3 or fewer and the edits preserve the first few letters of the given keyword, the user is prompted to check if the correct spelling was their true intention.

4.4 WordNet Synonyms

There are often multiple ways to communicate the same symptom, some of which cannot be captured using edit distance. Thus, we also use WordNet to generate synonyms for each valid symptom. Specifically, we look at the words in the “synset” of a symptom, their lemmas, and their derivation-

ally related forms. This allowed us to group together noun, verb, and adjective versions of the same symptom and to match phrases like “throw up” to symptoms like “vomiting”, which are synonyms that are very different in terms of edit distance.

For multi-word symptoms that are difficult to find synonyms for (e.g. “ache all over”), we generate permutations of the words in the symptom and try removing suffixes like “ing” and “ness” to see if any of the modified symptom phrases match a substring in the user input. Both permutations and synonyms from WordNet synsets were added as synonyms for Wit.ai symptom keywords, so if a user inputs “throw up”, Wit returns the entity “vomiting”, which is a valid ApiMedic symptom.

4.5 Cosine Similarity

We also use cosine similarity of GloVe vectors to help map extracted symptom substrings to symptoms in ApiMedic. We express the raw extracted symptoms and the ApiMedic symptoms as sums of word vectors (excluding those for stop words). This helps to generate candidate symptoms that we then ask the user to choose from or confirm. For example, Wit sometimes extracts phrases like “vision is bad”, which would originally be ignored because they do not contain all the words of any valid symptom. Using cosine similarity of GloVe vectors allows us to map these phrases to ApiMedic symptoms like “vision loss for far objects” (cos sim. = 0.836), which are unlikely to appear in user input with their exact phrasing.

5 Evaluation

Due to the nature of conversational agents, their evaluation is inherently subjective, but processes have been developed to formalize their assessment, such as the PARADISE methodology. The PARADISE methodology is built on the idea that user satisfaction is maximized when the task is correctly completed and costs are minimized. (Walker et al., 1997) Cost minimization is usually measured by a combination of conversation efficiency and qualitative measures. Unlike in applications such as flight planning, where there are set number of slots to be filled, there isn’t a fixed number of symptoms we were expecting, and longer interactions could signal increased interest. Therefore, our evaluation focused on slot filling statistics and qualitative measures.

5.1 Data

In order to get quantitative results, we needed a data set of symptom descriptions in free text. Since there is no such data set that publicly exists, we sent out a survey to collect examples. The survey asked for demographic information, a natural language description of symptoms, further elaboration on the symptoms, and the presumed diagnosis. We received 59 responses from participants aged 20 to 60+, leading to a diversity in response styles. To prevent bias in the construction of our workflow, we went through the completed surveys after the iterative improvements to the symptom extraction pipeline were completed. By hand, we extracted the symptoms from the descriptions, mapped these to valid symptoms from the ApiMedic database (if possible), and then mapped the proposed diagnosis to the corresponding diagnosis in the ApiMedic database (if it existed). One such survey response and the corresponding workflow are shown and described below:

What symptoms are you experiencing?
“I have some congestion in my nasal cavity.”
Can you describe your symptoms in more detail?
“There’s a lot of phlegm and it’s really uncomfortable.”
Did you have a specific diagnosis in mind?
“Cold.”

For this example, we then manually extracted the symptoms “congestion” and “phlegm”, manually mapped these to valid ApiMedic symptoms, “28: Nasal congestion” and “64: Sputum”, and manually mapped the proposed diagnosis to a valid ApiMedic diagnosis, “80: Cold”.

5.2 Quantitative Evaluation

We evaluated the system end-to-end on its ability to diagnose the conditions in the aforementioned data set based on the demographics and symptom descriptions inputted by survey participants. Performance on the symptom extraction and symptom mapping tasks were quantified using precision and recall, and diagnosis performance was evaluated through accuracy.

In addition, to testing the system end-to-end, we additionally tested each of the three main tasks in our pipeline (symptom extraction, symptom mapping, diagnosis) in isolation, so that we could pin-

	Symptom extraction	
Recall	64%	
Precision	66%	
	Symptom-code mapping	
	With bot-extracted symptoms	With hand-extracted symptoms
Recall	64%	66%
Precision	72%	78%
	Diagnosis accuracy	
	With bot-mapped symptom codes	With hand-mapped symptom codes
Accuracy	34%	37%

Figure 3: Slot-filling results for symptom tasks.

point any bottlenecks in performance. To test each task in isolation, we evaluated its performance by feeding the ideal, handcrafted output from the previous step into that task. The results from this evaluation can be seen in Figure 3.

There was perfect slot filling for both age and gender, since there was a very restricted number of ways people provided those two pieces of demographic information. Therefore, all of our analysis focuses on the symptom and diagnosis tasks.

We see that approximately 2/3 of symptoms were extracted, and approximately 2/3 of those extracted were in fact present. False positives often stemmed from a misunderstanding of words with multiple meanings, or that require context. For example, “heavy” was supposed to be specifying the extent of a symptom, but it instead mapped to the “overweight” symptom. In another case, a woman discussing menstruation mentioned “irregular spotting and bleeding”, which was extracted just as “bleeding”, which misrepresents the nature of the situation. Some additional false positives stemmed from Wit.ai, which assumed “pain and” was a misspelling of “hand pain”. The Wit.ai API doesn’t map its extractions to the original text, which made it difficult to reduce that sort of error without reverse engineering the process. False negatives often stemmed from symptoms that were described unconventionally, e.g., “my heart is all fluttery”, and are therefore difficult to resolve without a more complex model.

Symptom mapping also had recalls in a similar range to symptom extraction in both cases where input symptoms were bot- or hand-extracted, but precision was higher since there was a constrained set of possibilities, which got rid of many false

positives from the previous section. We can see here that the numbers do not indicate much of a drop-off in performance in the symptom mapping step. This can primarily be explained by the fact that Wit.ai primarily found symptoms that were easy to map. An example of a symptom that was unable to be mapped was “blindness” to “vision loss”.

Finally, in terms of diagnoses, we see that accuracy was 34%. However, we notice that this is not that much lower than the accuracy of 37% one would have if the symptoms were perfectly mapped and extracted. This can partially be attributed to the fact that via the survey, patients could not answer additional questions available in the live bot that could help the agent narrow down the possible diagnoses. Multiple diagnoses can share the same few symptoms, but, in general, most of the bot-generated diagnoses sounded reasonable.

5.3 User Feedback

For our qualitative evaluation, we focused on the questions posed by the PARADISE methodology.

Our testers found that it was clear what kinds of input they needed to give and the follow-up questions were well-directed. They also felt like it was evident what the bot understood and processed their inputs correctly.

Although there sometimes were some hiccups with the bot’s natural language understanding, they found that upon being asked to rephrase, the bot was able to understand the intention. There were no latency problems detected with the time needed for the bot to respond.

Given more fine-tuned symptom detection, one user said that it would be very easy and effective to continue using the bot.

6 Conclusion

We succeeded in building a simple medical chatbot that provides personalized diagnoses based on symptoms. In the future, the bot’s symptom recognition and diagnosis performance could be greatly improved by adding support for more medical features, such as location, duration, and intensity of symptoms, and more detailed symptom description. This bot uses an external, closed-source diagnosis engine, so in order to improve the detail and quality of the diagnosis it may be necessary to develop an engine from scratch or find another

resource that supports expansion.

The top symptom candidates as determined by cosine similarity were not always good suggestions. For example, the top five most similar symptoms to “sweat” using this measure are: “tears”, “perspiration”, “sweating”, “coughing up blood”, and “yellow colored skin” (in decreasing order of cosine similarity). While “perspiration” and “sweating” are exact synonyms for “sweat”, the other candidates are relatively unrelated. We used pre-trained GloVe vectors trained using word-to-word co-occurrence, so it makes sense that “sweat” and “tears” would be most similar because “blood, sweat, and tears” is a common phrase. Training GloVe vectors on a medical corpus could help with this issue.

Currently, we only have light progressive prompting, but in the future, we would hope to implement more structured progressive prompting when the bot is unable to understand user input. This, for example, could be done by asking the user to break down their symptom by first asking for the body part they are referring to to help narrow down the list of possibilities. Although we did attempt to implement body part extraction, the mapping from body part to symptom was rather coarse and would require some hand-labeling to do well, which we chose not to focus on.

Another future addition is allowing the agent to recommend treatments, which could easily be done by integrating a database of illnesses and associated treatments.

Hopefully, future automated medical agents will quickly, reliably, and cheaply diagnose patients, recommend treatments, and even prescribe medication. This could simultaneously reduce costs and increase convenience for patients as well as reduce the burden on overworked doctors.

7 Appendix: Example Chatbot Conversations

Note the suggested symptoms and mapping of “throw up” to “nausea”.

```
Doc>
      .-----.
      ==(_)==  Welcome!
      // 6 6 \\ /
      ( 7 )
      \ '._' /
      _.'_. /
      --) (---

Hello! I'll be your doctor today.
> Hi!
Doc> Okay, let's see what's up with you. Ready?
> Okay
Doc> What is your age?
> I am 33
Doc> What is your sex?
> woman
Doc> What brings you to my office today?
> I've been needing to throw up a lot lately.
Doc> Sorry to hear about your vomiting. Some other common symptoms people also experience with your profile include diarrhea, menstruation disorder, and missed period. Do you have any of those, or any other symptoms?
> Yes, I have missed my period lately.
Doc> Sorry to hear about your missed period. Are you experiencing any other symptoms?
> No!
Doc> You most likely have pregnancy. Another possibility is pregnancy-related problems. I would recommend you follow up with a doctor who specializes in gynecology. Are you having any other issues you want to talk through?
> No thank you :)
```

Note the variation in age and sex inputs.

```
> Hello!
Doc> I'm going to ask you some questions. Coolio?
> Okey dokey.
Doc> Uh, okay... What is your age?
> I am twenty-two.
Doc> Are you male or female?
> I'm a guy.
Doc> What has been bothering you?
> I feel extremely nauseous and have a really bad fever.
Doc> Sorry to hear about your fever and nausea. Along with those, people often also have diarrhea, stiff neck, and runny nose. Do you have any of those, or any other symptoms?
> No, I don't...
Doc> You most likely have stomach flu. It could also potentially be food poisoning. I would recommend you follow up with a doctor who specializes in internal medicine. Would you want to try diagnosing something else?
```

Note the variations in templates and the option to begin a new diagnosis session at the end.

```
Hello! I'll be your doctor today.
> Hihi :)
Doc> I have a few questions for you. Okay?
> Sounds good
Doc> How old are you?
> I'm 55
Doc> Oh, getting along in the years now, aren't you? Are you male or female?
> Hey! ... I'm a guy
Doc> What brings you to my office today?
> I've had a really bad cough for the last few months.
Doc> Sorry to hear about your cough. I've noticed that patients I've seen with h your symptoms also tend to suffer from stuffy nose, sore throat, and nosebleed. Do you have any of those, or any other symptoms?
> Yeah, my nose is pretty stuffy.
Doc> Sorry to hear about your stuffy nose. Any more symptoms?
> No
Doc> You most likely have cold. It could also potentially be flu. Are you having any other issues you want to talk through?
> |
```

Note the suggestion of “blurred vision” based on the single keyword “blurred”.

```
Doc> I have a few questions for you. Okay?
> Okay
Doc> How old are you?
> twenty
Doc> What is your sex?
> I'm a woman
Doc> What has been bothering you?
> Things just seem a bit blurred
Doc> When you say blurred, do you happen to mean blurred vision?
> Yes, that's right
Doc> Ok. Some other common symptoms people also experience with your profile include vision impairment for near objects, vision impairment, and sensitivity to glare. Do you have any of those, or any other symptoms?
> hmmm... I'm definitely experiencing some vision impairment
Doc> Sorry to hear about your vision impairment. Any other symptoms been bothering you?
> No
Doc> You most likely have far-sightedness. It could also potentially be green cataract. I would recommend you follow up with a doctor who specializes in ophthalmology. Would you like to talk through your illness again?
```

References

A. Bhargava, A. Celikyilmaz, and D. Hakkani-Turand R. Sarikaya. 2011. Easy contextual intent prediction and slot detection .

CDC. 2015. Access to health care.

Dylan Furness. 2016. The chatbot you will see now.

Jelor Gallego. 2016. An ai-powered chatbot is helping doctors diagnose patients.

Sean P. Keehan, John A. Poisal, Gigi A. Cuckler, Andrea M. Sisko, Sheila D. Smith, Andrew J. Madison, Devin A. Stone, Christian J. Wolfe, and Joseph M. Lizonitz. 2016. National health expenditure projections, 201525: Economy, prices, and aging expected to shape spending and enrollment. *HealthAffairs* pages 1522–1531.

Priaid. 2016. *Apimedic*. <https://apimedic.net/>.

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. Paradise: a framework for evaluating spoken dialogue agents. *EACL* pages 271–280.

Puyang Xu and Ruhi Sarikaya. 2013. Exploiting shared information for multi-intent natural language sentence classification. *Microsoft*.

Nicole Yankelovich, Gina-Anne Levow, and Matt Marx. 1995. Designing speechacts: Issues in speech user interfaces. *CHI* pages 369–376.