Neurological Disease Diagnosis with Watson

Kevin Neilson, Marissa Le Coz, and Evan Honnold

November 14, 2016

1 Introduction

We used IBM's "Watson" cognitive-computing services to create an application for diagnosing neurological disorders. The application interacts with the user through a chat-bot interface: it poses several questions about the user's symptoms, the user types their answers in natural language, the app processes the user's answers, and then a list of diseases consistent with those symptoms is presented to the user. If no relevant diseases are found, we tell the user - and we make it clear that this application is narrow in scope and should not be used for diagnoses and treatment decisions in its current state.

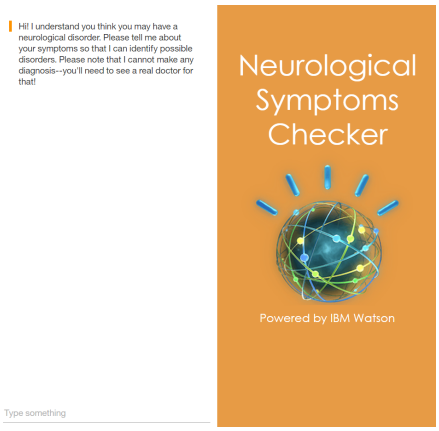

Figure 1: GUI at Start of Conversation

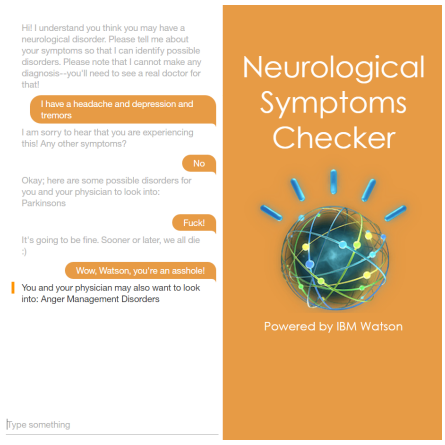

Figure 2: GUI at End of Conversation

2 Why Use Cognitive Computing?

Our inspiration for this project came from Dr. Filiano, a neurologist at DHMC. Marissa met Dr. Filiano when he came to the DALI Lab with the idea of making his book on rare neurological diseases available in app form (by taking symptoms as input and outputting possible diseases). He had placed in DEN's "The Pitch" with this idea. Marissa was only on the project for part of a term and did not know what had become of it. Upon taking CS89, she thought Watson would be the perfect tool to turn his idea into a reality, so she contacted him to ask what DALI had produced.

Due to the complex information in Dr. Filiano's book, the DALI Lab was only able to categorize diseases based on one symptom. This required developers to read the book and make a complex flow chart for predicting diseases. Cognitive computing is an overall better approach. First, because Watson can understand unstructured data, it can "learn" Dr. Filiano's book without humans needing to make complex flow charts. Second, what if Dr. Filiano updates his book? Hard-coded flow charts are a pain to alter. Watson, on the other hand, can easily train on new versions of the book. Further, Watson can process queries much more quickly and efficiently.

Cognitive computing solves a number of other problems as well. The natural language processing allows the app to accept information in whatever form the user provides it. For example, the user can say "Why does my head hurt?" or "My head aches" or "I have a headache," and in each case Watson will recognize that they are referring to the "headache" symptom.

Ultimately, we ended up not using Dr. Filiano's book due to his copyright concerns. (He wondered, what would happen to the proprietary information in his book once Watson trained on it? Would IBM "own" it? Professor Palmer looked into this question and found out that IBM does not have any official copyright information for Retrieve and Rank.) Instead, we trained Watson on the content of freely available webpages about some more common neurological diseases. We intend our app to serve as a prototype for the sorts of things Dr. Filiano could do with Watson in the future when IBM has a clearer copyright policy.

3 Implementation Details

The three IBM Watson services we used were the Watson Conversation Service, Document Conversion, and Retrieve and Rank. We divided the work of the project equally: Marissa developed the conversation service and front end web component, Evan handled the client-server interaction of the chatbot with the other Watson services, and Kevin developed and trained the Retrieve and Rank Service.

The basic flow of the application is as follows: The user navigates to our the web application and is greeted by a ChatBot interface that asks the user for their symptoms. The user then, in NLP, conveys their symptoms, which are identified as entities in the Watson conversation service, and then passed to the Retrieve and Rank service via an HTTP request. The Retrieve and Rank service returns the documentation of the disease most closely related to the symptoms entered. The app.JS javascript file then parses this JSON response and informs the user in NLP whether their symptoms match a neurological disease, if so, recommends that they seek the advice of a medical professional.

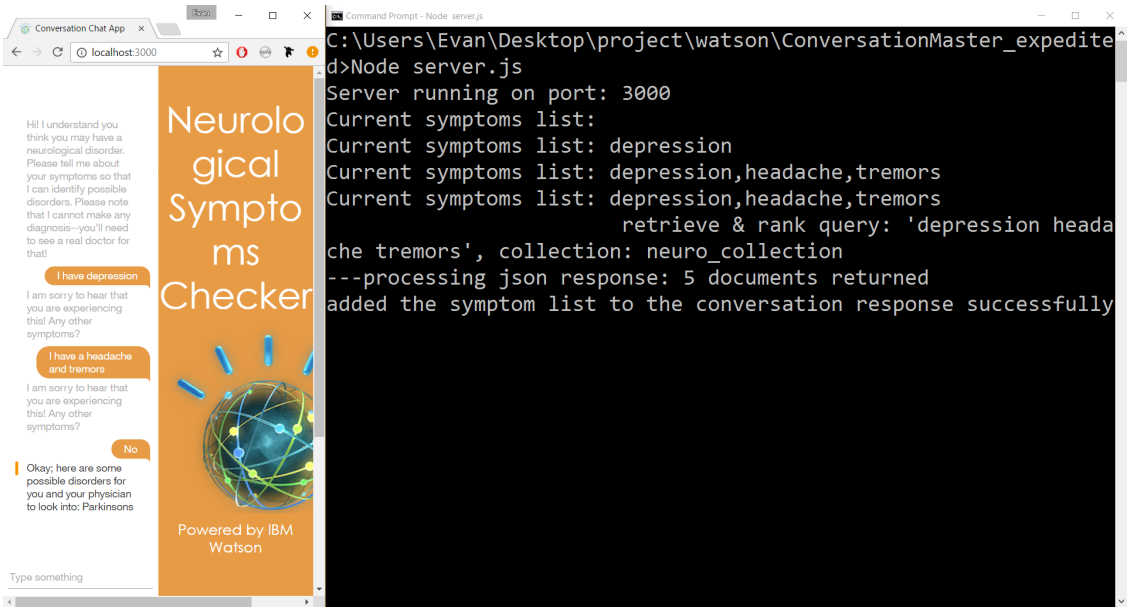

Figure 3: Console Output to Demonstrate Steps of Operation

We faced some challenges while setting up and training the Retrieve and Rank service. We started using curl commands to set up the service and switched to Watson's web interface but returned to using curl commands for the increased flexibility and customization it gave us.

Watson rejected our first few trainings because of an insufficient number of feature vectors, and then an insufficient number of queries in the ground truths file. We were able to update the ground truths file with a total of seventy-one queries spanning the three neurological diseases.

We also faced some challenges when saving the list of symptoms the user inputs. The chatbot operates in a simple loop: it asks the user for symptoms, the user gives one or more symptoms in natural language (with the intent $\#symptoms$, i.e., "I have", "I feel", etc.), the chatbot asks if there are any more symptoms, the user gives one or more symptoms in natural language, etc. When the user says, "no," (which is defined as the intent $\#no$) that there aren't any more symptoms, Retrieve and Rank uses the chatbot's list of symptoms to give potential diseases, if any. Watson offers the construct of "context variables" to save values from input to input. Context variables can be altered in the "Advanced" section of a dialogue node. Initially, we set up list (as a context variable) to which new symptoms were appended. This worked fine as long as the user only gave one symptom at a time. When the user gave multiple symptoms, these were accessible as $entities.symptoms[0].value, entities.symptoms[1].value, ..., entities.symptoms[n].value$ for n symptoms. From the documentation available online, it appears that it is impossible to append a variable number of items to a context variable list from the dialog node interface on Bluemix. Surprisingly, "$symptoms\_list$" : " $<?context.symptoms\_list.append(entities.symptoms[@].value)?>$ " did not work, nor did embedding a for-loop. To solve this problem, Evan created a global variable in the Javascript file to which symptoms are appended. Given how well this works, it is curious why "context variables" are a construct in the first place.

4 The Business Plan

The target user of our application is an average person with limited medical knowledge-either the sick person or someone worried about him or her. The symptoms recognized by our system are everyday terms such as "headache," "dizziness," and "double vision." Dr. Filiano's book likely contains more technical symptoms, such as the levels of certain hormones in the patient's blood, something the average person would not be able to measure. If we were to train our application on Dr. Filiano's book, the target user would likely be doctors who do not have specialized knowledge of rare neurological disorders instead of everyday people.

In the case of the web application we created, a good business plan would to make the

web application available on the Internet, subsidized by ads. In the case of an application with more technical symptoms requiring medical knowledge, a good business plan would be to sell subscriptions to the web application to hospitals for doctors to use. This would enable doctors without Dr. Filiano's specialized domain knowledge to possibly identify and treat more disorders more quickly.

5 Assessment of Results

Our application successfully executes its intended purpose - to be able to identify neurological diseases. The application has performed as expected in a number of test cases, but we recognize that it's limited by the extent of the training data and NLP processing. Our application can handle multiple symptoms and should the user enter symptoms that match more than one neurological disease, the application will return all of those matching diseases.

6 Areas of Improvement

We know a number of ways that we would improve and expand the capabilities of the application. Our Retrieve and Rank Application is able to understand queries in NLP, and we established the training framework for it to be able to interpret follow-up questions, such as "How many people per year are diagnosed with ALS?" or "Tell me more about ALS."

Our application can identify the symptoms of three distinct neurological diseases: ALS, Parkinson's Disease, and Giant Cell Arteritis. An expanded version of this application would be able to identify more neurological diseases, as there are quite a number of them.

Another way to improve our application would be by enabling users to send a report to their physician from the application, summarizing the symptoms they have inputted, Watson's disease predictions, and excerpts from the corresponding Retrieve and Rank documents.

7 Conclusion

7.1 "Do you think it would succeed?"

Our motivations for this project stemmed from the fact that many neurological diseases go undetected, and individuals miss the opportunity for treatment when it is most effective. We wanted to lower the "barrier to entry" for gaining medical insights (i.e. asking Watson a few

questions is quicker and cheaper than a Doctor's visit) and help people who think they might be afflicted with a neurological disease get help. With the disclaimer that Watson and our project are not substitutes for professional medical advice, we believe that we have built an MVP (minimum viable product) of a cognitive computing medical adviser.

While we don't believe that cognitive computing will replace doctors, we absolutely think that cognitive computing can be used to provide medical assistance in situations where a doctor isn't needed or perhaps where one isn't available. We envision a future where access to basic medical knowledge is immediate and freely available to all. Imagine the benefits of medical chatbots or speech to text systems that can walk a human through providing first aid before first responders arrive.

References:

Special thanks to Charles Palmer, Armen Pischdotchian, and Dr. Filiano.
We developed our chatbot by modifying Armen's tutorial code.

Data Sources:

http://www.webmd.com/parkinsons-disease/guide/parkinsons-common-symptoms
http://www.webmd.com/parkinsons-disease/guide/understanding-parkinsons-disease-symptoms
https://www.ucsfhealth.org/conditions/als/index.html
http://www.alsa.org/about-als/symptoms.html?referrer=https://www.google.com/
https://www.mda.org/disease/amyotrophic-lateral-sclerosis/signs-and-symptoms
https://en.wikipedia.org/wiki/Giant-cell-arteritis
http://www.mayoclinic.org/diseases-conditions/giant-cell_arteritis/basics/definition/con-20023109
https://www.ucsfhealth.org/conditions/parkinsons_disease/signs_and_symptoms.html