Robyn: A Natural Language Interface to Database System for Medicine

Ashutosh Rai
Earlham College
801 National Rd W
Richmond, IN, 47374
arai13@earlham.edu

ABSTRACT

Found in various platforms such as smartphones, computers, web platforms and popular commercial software, natural language interface to database (NLIDB) systems are ubiquitous in today's world. Despite the rapid growth of the field, the limited availability of resources to learn about building such systems creates a high entry barrier. Additionally, scarcity of a reliable NLIDB system that is dedicated to medical health, which is a topic of uttermost importance, persists as well. Robyn is a web application that attempts to solve both these issues. It is built with easily accessible software tools and architecture. Robyn uses AIML as the NLP engine, SQLite as the database engine, and Python to put all the components together as well as to provide the interface for the users. Bottle, a micro web-framework written in Python, is used to host Robyn on the server side while HTML/CSS and JavaScript is used for Robyn's web interface.

Keywords

Natural Language Interface to Database; Natural Language Processing; AIML

1. INTRODUCTION

A natural language interface to database (NLIDB) is a system that allows the user to access information stored in a database by typing requests expressed in some natural language (e.g. English)[1]. It allows technical actions to be activated without the rigidity of technical sentences and syntaxes, which is replaced essentially by a barrierless natural conversation. Modern Intelligent Personal Assistants (IPA) like Siri, Cortana and Google Now could be considered some popular instances of such a system. However, these are general purpose programs that complete tasks and answer questions related to a range of fields rather than focusing on a particular field. This paper will present Robyn, an NLIDB system with medicine as the domain of knowledge.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WOODSTOCK '97 El Paso, Texas USA
© 2016 ACM. ISBN 123-4567-24-567/08/06...$15.00
DOI: 10.475/123_4

While Intelligent Personal Assistants (IPA) and chatbots were exciting, new, and relatively unfamiliar phenomena a decade ago, they have since become a household tool used in everyday life. These tools can be used to accomplish a wide array of tasks such as checking the weather, booking movie tickets, sending messages and completing financial transactions. The question may then arise: what is the purpose of building an NLIDB system that has a narrow focus like Robyn when other systems like Siri can accomplish a plethora of tasks along with answering questions, including the ones related to the narrower knowledge domain of focused NLIDBs itself? The answer lies in the question itself. While both could answer the same question - "What are the symptoms of Tuberculosis?", the difference would lie in the quality of answers that would be returned. Whereas a general purpose IPA would query the question in Google and then return the search results, a system such as Robyn would be able to provide high quality and reliable answers that would not require the user to filter the search results to find the answer. Accessibility, efficiency and reliability are key.

Python, AIML and SQLite were primarily used to build Robyn. AIML (Artificial Intelligence Markup Language) is an XML-compliant language used for creating natural language software agents. It was used to handle the natural language processing for the interaction. A SQLite backend was used for the database, while Python was used to put all the tools together, provide an interface and handle the queries. Various designs for natural language processing, storing the database, handling queries and putting together the different tools were explored before a hierarchical pattern based design was implemented. Python handled the queries, setting up the statements, communicating with the SQLite database and then returning the results to the user. Limited context is maintained as well, using AIML to store the topic of the current conversation (e.g. which disease). Bottle was used as the web framework to host Robyn on the server-side.

This paper discusses how Robyn was built, going from a bigger picture of NLIDBs to the very specific details of the tools and algorithms used for Robyn. In addition to exploring NLIDBs, including their history, general techniques and architecture of Robyn, this paper also presents the implementation of the software and the testing of Robyn. To accomplish these goals, several aspects of the system are examined in the paper. Section 2 presents several related work that influenced Robyn and the architecture, while section 3 and 4 analyze the design and implementation details

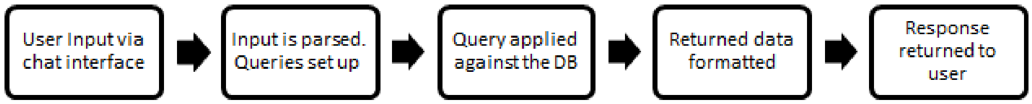

Figure 1: General workflow for NLIDB systems

of Robyn. Section 5 discusses results of where Robyn stands currently and section 6 finally closes with a conclusion and the identified future goals.

2. RELATED WORK

The late sixties and early seventies saw natural language interface to database systems appear on the scene for the first time[1]. These systems were designed with very specific databases in mind so the systems could not be easily adapted to a different database. Today, all aspects of such systems have progressed greatly. Natural language processing has improved to produce a more realistic human-like conversation, systems are more adaptable and portable, and such systems have been integrated into other convenient areas like Facebook messenger and Google. However, it is still an active research field with much more to be done. This section will shed some light on the history of NLIDBs and the design of such systems.

2.1 Brief Background of NLIDB Systems

The best known NLIDB in the early period was Lunar, a natural language interface to a database containing chemical analyses of moon rocks[8]. Early NLIDB systems such as Lunar were each built having a particular database in mind, which made them difficult to use with different databases. By the late seventies, several more NLIDBs had appeared, such as Rendezvous and Ladder, which could be used with large databases, and could be configured to interface to different underlying database management systems[1].

In the mid-eighties, numerous NLIDBs were developed that demonstrated impressive functions and potential. However, NLIDBs did not gain the rapid and wide commercial acceptance that one would expect after seeing the progress of the field. The development of successful alternatives to NLIDBs, like graphical and form-based interfaces, and the intrinsic problems of NLIDBs were probably the main reasons for this stunt of growth[1]. However, the growth of NLIDBs has been greatly accelerated lately with improvements in speed, efficiency, portability and adaptability. With the growth of IPAs such as Siri and Cortana, as well as the integration of chatbots with access to data in commercially successful platforms like Facebook, it looks like NLIDBs are here to stay and further grow.

2.2 Basic Architecture

Generally, there is a common flow of tasks that most NLIDBs follow. While the specific tools and algorithms may differ, a higher level pattern can be observed. Some aspects of that high level analysis of the basic architecture are described in this section.

2.2.1 The Bigger Picture

Most NLIDBs, including IPAs such as Apple's Siri, follow the following workflow to accomplish a task:

1. Abstract the input text from the interface and pass it on to the next layer. If the system supports speech then transcribe human speech such as commands, questions, or dictations into text before passing it.

2. Use natural language processing to translate transcribed text into "parsed text". The parsed text is then analyzed, detecting user commands and actions. ("What is Bryan's contact number?", "What is the weather like?", etc.)

3. If the system is linked to a specific database, apply relevant queries to extract the necessary data. In cases of general purpose systems, data mashup technologies are used to interact with 3rd-party web services to perform actions, search operations, and answer questions

4. Transform output of the database or 3rd-party web services back into natural language text.

5. Present the output to the user interface. Use TTS (text-to-speech) technologies to transform the natural language text into synthesized speech if necessary.

This is a common workflow that many NLIDB systems implement, including Robyn. However, as Robyn has a very specific domain of knowledge, setting up appropriate queries and applying them against specific databases remains central in the workflow for Robyn. The relationship can be seen diagrammatically in Figure 1.

2.2.2 Layers of an NLIDB

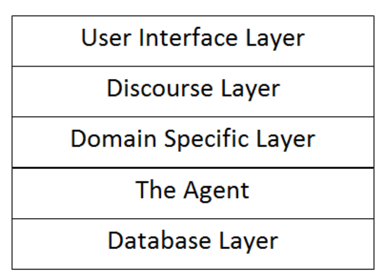

Figure 2: Bhatia's Layered Architecture

Bhatia et al. present a higher level architecture that represents most standard NLIDB systems found today. They describe their architecture based on an n-tier layered architecture as described below[2]:

- The first layer is the User Interface (UI) layer which has the user interface module. The user interface provides a communication interface to interact with the user. The user interface module gives the input sentences to the Dialogue Manager which parses the sentence and identifies the intention of the user.

- The second layer is the Discourse layer. This layer performs the actions of the Dialogue Manager which are helpful in identifying the intention of the user.

- The third layer is the Domain Specific layer. This layer requires the domain specific knowledge to be encoded in the Dialogue Manager.

- Once the intention is identified, it makes a call to an appropriate agent. The agent forms the fourth layer of the server. The agent performs the required action and passes the results back to the Dialogue Manager.

- The fifth layer is the Database layer, which represents the knowledge of the application.

Based on Bhatia's et al.'s model, Robyn implements the layered architecture. While the layers are not explicitly mentioned in Robyn, the model is very evident in the architecture.

2.2.3 *Dialogue Model*

An important part of the architecture of any NLIDB is the handling of dialogue. To have a meaningful and efficient conversation, the Agent Based Dialogue Model mentioned by Wobcke et al. is an informative as well as intuitive method. A simple layout of the model is present in figure 3 [7]:

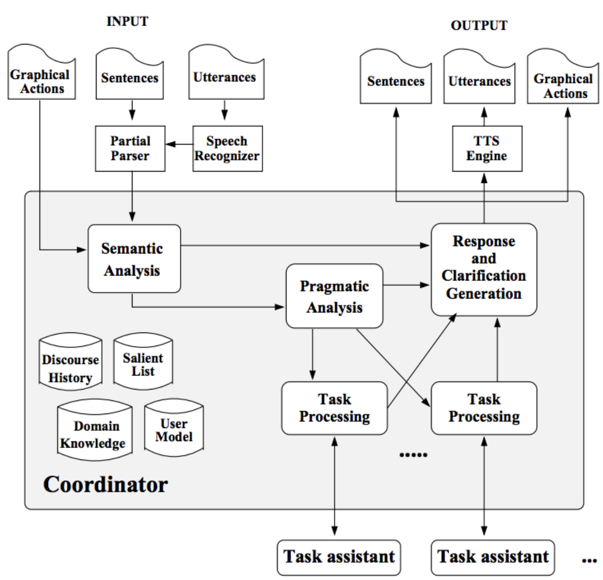

Figure 3: Agent-Based Dialogue Model[7]

The Coordinator maintains the dialogue model, including the conversational context and other domain-specific knowledge as its internal beliefs, as follows:

- Discourse History: maintains the conversational context such as information about the current and past dialogue states

- Salient List: maintains a list of objects which have been mentioned previously in the conversation, i.e. the objects that are in the focus of attention

- Domain-Specific Knowledge: includes domain-specific vocabulary and information of the tasks that are supported, used in interpreting the user's requests

- User Model: maintains information about the user such as current device, preferred modality of interaction, physical context, preferences, etc.

While the natural language processing aspect of Robyn in its current state is still in the early stages, several aspects of this agent-based model will be implemented in the near future for dialogue handling and context maintenance.

3. DESIGN

This section describes the different aspects of that design such as a higher level analysis of the system, software used, and the architecture.

3.1 Higher Level Analysis of Robyn

Three broad goals were identified that would be necessary to construct an NLIDB system:

- Building a chatbot interface that handled the conversation and understood the requests/questions from the user. This would be the natural language processing part of the system.

- Creating/compiling a database that had reliable information related to the domain of knowledge, medicine in this case.

- Integrating the database into the chatbot and setting it up so that the information can be accessed in an organized, efficient manner.

While the above goals were necessary to create a system like Robyn, Robyn also required a workflow. Robyn, at a high level, implemented the models discussed in sections 2.2.1 and 2.2.2, ensuring the system would function. The workflow in Robyn works like the following:

1. First, via a text based interface, Robyn provides a platform for users to communicate and ask questions in a natural language (English, in this case). The input could be general phrases such as "Hello!", "How are you?", or questions related to medical health, in which Robyn specialized such as "What is AIDS?" or "What are the symptoms of common cold?"

2. The input would then be analyzed and parsed. If it is a general phrase such as "How are you?" then an appropriate response such as "I'm fine, how about you?" would be chosen. However, if the input is a medicine related question then a necessary query is formed.

3. The query then would be applied against the right database.

4. After the correct data is extracted from the database, the data is then put in a sentence in a natural language (English for Robyn), maintaining the flow of the natural language.

5. Finally, the response, whether general phrase or medicine specific, is then output in the text based interface

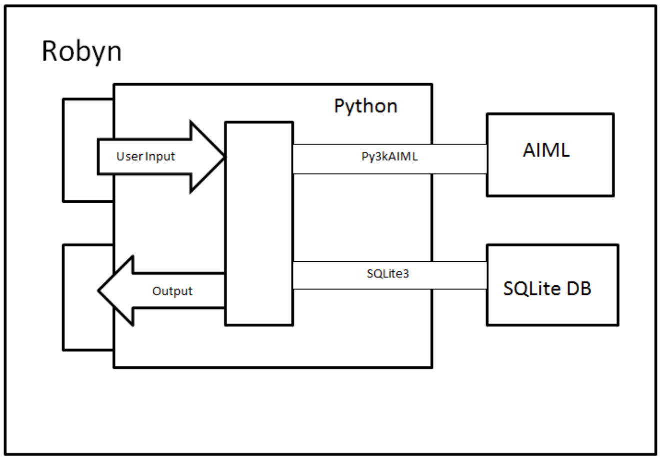

Figure 4: Robyn's software architecture

3.2 Software Used

Brief descriptions of the software used, and the reason for the choice are given below. The details about the implementation are discussed in section 4. The relationship of the software used can be seen in the architecture depicted in figure 4.

3.2.1 AIML

AIML stands for Artificial Intelligence Markup Language. It is an XML-compliant language used for creating natural language software agents. It can be used to handle natural language processing, and provides tools to do so. AIML describes a class of data objects called AIML objects and partially describes the behavior of computer programs that process them[5]. AIML objects are made up of units called topics and categories, which contain either parsed or unparsed data. Each category tag contains a pair of pattern and template tag. The system searches the pattern according to user's input, and produces the appropriate template as a response[6].

AIML is the heart of natural language processing part of Robyn. When the user inputs a text, AIML handles the response and provides the necessary query that is applied against the database if necessary. AIML does not interact directly with the UI layer, but rather talks to the Python program, receiving the user input from the Python program and sending the response/query statements back to it.

3.2.2 SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine[4]. SQLite is an embedded SQL database engine but unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file.

The portability, low barrier to accessibility and compactness of SQlite made it an excellent choice as the database engine for Robyn. Additionally, thanks to solid reliability of the SQLite3 module in Python, the process of integration of the SQLite database was easy and reliable.

3.2.3 Python

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language whose design philosophy emphasizes code readability. While Python is an excellent tool in general, what made it perfect for Robyn was the availability of modules and the smoothness of the implementation of these modules. This made the integration of the database as well as the natural language interpreter quick as well as reliable.

The two main modules that were used in the Python scripts were Py3kAIML[3] and SQLite3[4]. Py3kAIML is an interpreter for AIML, implemented entirely in standard Python. It allows for the Python program to interact with the AIML scripts. The SQLite3 module provides a SQL interface compliant with the DBAPI 2.0 specification. It allows the Python program to connect to the database and perform database operations.

3.2.4 HTML/CSS/JS + Bottle

HTML/CSS and JavaScript (JS) are standard choices of tools for most websites and web applications. HTML provides structure to the webpage, while CSS gives the styling to the graphical aspect. Finally JavaScript helps to add dynamic functions to the page.

HTML/CSS and JavaScript were used for Robyn's web interface to design a chat box. Furthermore, as Robyn is primarily written in Python, Bottle was used to host the scripts on the webpage. Bottle is a fast, simple and lightweight WSGI micro web-framework for Python, which was perfect for Robyn.

4. IMPLEMENTATION

The implementation of Robyn included of several independent components that came together to provide the functionality. Described below are some of the aspects of the implementation.

4.1 Connecting Robyn with AIML/Database

Python provides reliable, minimal configuration modules for AIML as well as SQLite. Hence, integration of AIML and the database took minimal coding. While Robyn's natural language processing is handled by multiple AIML files, they are all linked through a single standard startup file, which makes the necessity to edit the Python code minimal when the AIML file structure has to be changed.

4.2 Dialogue Handling

On a basic level, AIML responds to user inputs by matching the pattern, choosing the one that is higher in the hierarchy in case of multiple matches.

Analyzing the code in Figure 5, when a user asks "How are you?" then the pattern is matched and AIML sends the response "Thanks for asking. I'm good, how are you?" to the

```
<category>
    <pattern>HOW ARE YOU</pattern>
    <template>
        Thanks for asking. I'm good, how about you?
    </template>
</category>
```

Figure 5: Example of AIML code for general responses

Python program. However, when a medical health related question is asked, appropriate data needs to be extracted from the database. Hence, a query statement needs to be returned by the AIML engine. Let's look at another example in Figure 6.

```
<category>
    <pattern>WHAT ARE THE CAUSES OF *</pattern>
    <template>
        !SELECT description FROM facts
        WHERE (id = (SELECT id FROM disease WHERE name='<star/>'))
        AND (fact = 'Causes');
    </template>
</category>
```

Figure 6: Example of the AIML code for medical health related questions

When questions like "What are the causes of common cold?" are asked by the user, the AIML will match the pattern and return a query statement to the Python program. The related database here is contained in two tables. The first table contains disease names and their unique id numbers. The second table contains facts about the diseases such as symptoms, treatment, descriptions along with the unique id numbers of the diseases. Here, the query statement will first grab the id of the disease, matching against the name of the disease provided by the user. Then, via another query using the id number, the matching fact is picked and sent to the Python program by SQLite. There is an exclamation mark right before the query statement, which is explained in the next subsection.

4.3 Handling Query Statements

It has been briefly discussed how the AIML engine sends the response to the Python script. After receiving the response, the script needs to decide if the response needs to be directly output or a query against the database is necessary. The way Robyn handles it is by placing a special character (!) in front of query statements, allowing identification of the nature of the response..

```
response = k.respond(user_input)
if response[0] == '!':
    response = response[1:]
    cursor.execute(response)
    temp = cursor.fetchone()
    response = temp[0]
```

Figure 7: Algorithm to handle queries

Here, as seen in the algorithm in figure 7, when the special character (!) is detected in the start of the response returned by the AIML engine, first the character is removed. Then the query statement is applied against the database, extracting the necessary data and returning it to the UI screen.

4.4 Designing Web Interface

After the first four prior mentioned components were correctly implemented, Robyn took the form of a command line interface NLIDB system. However, for reasons of accessibility, ease of use, and efficiency, a web interface was created for Robyn, making Robyn a web app in the process.

Robyn is a single-page web app with a structure similar to a typical chatbot. A chat box was set up to display the conversation between the user and Robyn. The user input was abstracted via a text field, which is passed to the python scripts handling the AIML, via the Bottle framework. The response returned by the python script is appended to the chat box. A visual state of the web interface can be seen in figure 8.

5. RESULTS

Robyn currently exists as a locally hosted web application. The system is fully functional, with all the connections successfully plumbed and running. At the moment there are about 3 large AIML files to handle some general day-to-day conversations such as "Hello!", "How are you?" and "Who are you". There are also some specialized AIML files to handle Robyn-specific conversations such as "What is AIDS?", or "What are the causes of Tuberculosis?" The database currently consists of about 4000 diseases with facts related such as description, symptoms, causes, diagnosis and treatment.

5 test cases in the form of sets of questions were formed to test Robyn. Each test case focuses on different specific features of Robyn that is important to Robyn's functionality. For this paper, the 5 features that the tests focused on were Robyn's knowledge base, handling of variation in questions, handling of variation of diseases, context maintenance and change of context with topic.

The test case runs successfully, with all the responses from Robyn matching the expected results. Robyn was able to answer about different facts related to flu including definition, symptoms, causes, treatments and diagnosis. When the same questions were asked in a different lingual form, Robyn was still able to answer the questions. Multiple diseases could be included in the same conversation session. Finally Robyn was able to maintain context when talking about a specific disease, and also was able to change context correctly when the topic was switched. The test cases as well as the results in the form of conversation between the tester and Robyn is transcribed below for each test case.

5.1 Knowledge Base

The first test case is focused on the breadth and depth of Robyn's knowledge about diseases. A series of questions are asked to see if Robyn can answer about various facts related to a disease. The level of detail of Robyn's answers is also examined in this test. While for other test cases Robyn's responses have been condensed into a couple of words for the purpose of the paper's conciseness, followed by ...... to mark the answer's continuation, for this test case Robyn's full responses have been included to present the level of detail and information of the responses.

Figure 8: Robyn's web interface

Test case:

- What is flu?
- What are the symptoms of flu?
- What are the causes of flu?
- What are the treatments of flu?
- What is the diagnosis of flu?

Results:

You: What is flu?
Robyn: Influenza or flu is a viral infection that attacks the respiratory system including the nose, throat, bronchial tubes, and lungs. Though it is commonly called flu, influenza is not the same virus that causes diarrhea and vomiting.
You: What are the symptoms of flu?
Robyn: Common signs and symptoms of flu include fever over 101 F among adults and 103-105 among children, chills and sweats, headache, dry cough, muscular aches and pains, fatigues and weakness, nasal congestion, loss or appetite and diarrhea and vomiting among kids.
You: What are the causes of flu?
Robyn: Three strains of viruses cause the flu namely influenza A, B, and C. A is responsible for the deadly influenza pandemics around the world, which strikes every 10 to 40 years. Type B meanwhile can lead to smaller and more localized outbreaks while Type C has never been connected with any large epidemic. Type C is also a stable virus compared to A and B since they are constantly changing and new strains can appears regularly.
You: What are the treatments of flu?
Robyn: Flu treatment is usually just bed rest and plenty of fluids. In some cases though, an antiviral medication such as oseltamivir (Tamiflu) or zanamivir (Relenza) may be prescribed. Said drugs can treat both Types A and B of influenza and work by deactivating an enzyme that helps the virus grow and spread. Oseltamivir is an oral medication while zanamivir is inhaled through a device similar to that of an asthma inhaler. Side effects such as lightheadedness, nausea, vomiting, loss of appetite, and trouble in breathing are the common side effects of both medications.
You: What is the diagnosis of flu?
Robyn: A thorough physical exam is needed to diagnose the flu and a chest x-ray may be needed if pneumonia is suspected. Blood work may also be needed such as CBC, blood cultures, and sputum cultures. An antigen detection test meanwhile is the most common method of diagnosing the flu. If the flu is widespread in the community, no further testing will be required.

5.2 Handling of Variation in Questions

The second case is focused on the range of questions Robyn can handle. Instead of constantly asking in the form of "What are the" questions, the questions were asked in a variety of forms.

Test case:

- What is flu?
- List the symptoms of flu.
- What causes flu?
- How can I treat flu?
- How can I tell if I have flu?

Results:

You: What is flu?
Robyn: Influenza or flu is a viral ......[Answer continued]

You: List the symptoms of flu.
Robyn: Common signs and symptoms of F ......
You: What causes flu?
Robyn: Three strains of viruses cause the flu ......
You: How can I treat flu?
Robyn: Flu treatment is usually just bed ......
You: How can I tell if I have flu?
Robyn: A thorough physical exam is ......

5.3 Handling of Variation of Diseases

The third test case examines if Robyn can talk about multiple diseases in the same conversation. Questions related to three different diseases were asked in succession.

Test case:

- What causes bronchitis?
- How can I cure dislocated shoulder?
- What is AIDS?

Results:

You: What causes bronchitis?
Robyn: Bronchitis is often caused ......[Answer continued]
You: How can I cure dislocated shoulder?
Robyn: Putting your shoulder bones back ......
You: What is aids?
Robyn: AIDS means Acquired Immune Deficiency ......

5.4 Context Maintenance

The fourth test case tests Robyn's ability to interact in context and maintain it. The diseases were not explicitly mentioned after the first time.

Test case:

- What is jaundice?
- What are the symptoms?
- How can I treat it?
- What causes it?

Results:

You: What is jaundice?
Robyn: Jaundice is the yellowish ......[Answer continued]
You: What are the symptoms?
Robyn: The symptoms of jaundice are ......
You: How can I treat it?
Robyn: With the exception of the treatments ......
You: What causes it?
Robyn: When red blood cells get old ......

5.5 Change of Context with Topic

The final test case checks if the context changes when the topic of the conversation (disease) changes. 3 related questions to a disease were asked in context before asking another 3 contextual questions related to a new disease.

Test case:

- What is anemia?
- What are the symptoms?
- How can I treat it?
- List the symptoms of broken hip.
- What causes it?
- How can I tell if I have it?

Results:

You: What is anemia?
Robyn: Anemia comes from the ......[Answer continued]
You: What are the symptoms?
Robyn: Symptoms of anemia can be vague ......
You: How can I treat it?
Robyn: Treatments for anemia vary ......
You: List the symptoms of hepatitis.
Robyn: Initial features are of ......
You: What causes it?
Robyn: Viral hepatitis can be caused ......
You: How can I diagnose it?
Robyn: Viral hepatitis conditions ......

6. CONCLUSION AND FUTURE WORK

Natural language interface to database systems have been a field of massive interest in the realm of artificial intelligence since its invention. With the integration of such systems in popular commercial machines, like Apple's Siri and Microsoft's Cortana, as well as platforms, like Facebook it can be safely assumed that further development of NLIDBs will continue. Robyn, an NLIDB with medical health as its domain of knowledge, is another example of such a system. The software used to write Robyn such as Python, SQLite and AIML are popular tools, which will hopefully make the barrier to learning more about such systems minimal. After some improvement, Robyn will hopefully be a valuable reference to when talking about medical health.

The core components of Robyn have already been written and connected, and the web interface is fully functional with medical information from the database easily accessible with minimal effort on the user's end. Following are some of the tasks I hope to achieve in near future to improve the range, quality and efficiency of the conversations:

- Platform: While the web app works just like expected at the current state, I hope to add some more functionality to Robyn such as connecting with other users, notes, quick search tab and so on.
- Natural Language Processing: While natural language processing of a system is something that will always be improvable, I wish to significantly focus on this aspect of Robyn. Along with expanding the range of conversations she can handle, I also wish to work on building and maintaining more context related to the conversations. Indexing of the conversation to remember the

specific user history is something I am considering as well.

- Expanding the Database: While the current database is fairly extensive with 4000 diseases, I wish to expand the database to include more related information. Possible expansion I am thinking about is database related to medicines/drugs, diet information, healthy habits and so on.

7. ACKNOWLEDGMENTS

This work was possible through the guidance and support of my senior capstone advisers/mentors Dave Barbella, Xunfei Jiang and Charlie Peck. I would also like to thank the CS Department at Earlham for funding the project, and Adam Shwartz for his assistance with debugging. Finally, I would like to thank the chatbots.org forum community for their valuable discussions.

8. REFERENCES

[1] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(01):29–81, 1995.

[2] S. Bhatia and A. Bharambe. Architecture for an mobile-based intelligent personal assistant. 2009.

[3] https://github.com/huntersan9/Py3kAiml. Py3kaiml.

[4] https://sqlite.org/about.html. About sqlite.

[5] https://www.pandorabots.com/pandora/pics/wallaceaimltutorial.html. Aiml overview.

[6] D. Q. Nguyen, D. Q. Nguyen, and S. B. Pham. A vietnamese text-based conversational agent. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 699–708. Springer, 2012.

[7] W. Wobcke, A. Nguyen, V. H. Ho, and A. Krzywicki. The smart personal assistant: An overview. In *AAAI Spring Symposium: Interaction Challenges for Intelligent Assistants*, pages 135–136, 2007.

[8] W. A. Woods, R. M. Kaplan, and B. Nash-Webber. *The lunar sciences natural language information system: Final report*. Bolt, Beranek and Newman, Incorporated, 1972.