Knowledge-based Disease Recommender System

1st Mingrui Han
Computer Science Department
George Mason University
Fairfax, VA, USA
mhan8@gmu.edu

2nd Ruchita Garde
Computer Science Department
George Mason University
Fairfax, VA, USA
rgarde@gmu.edu

3rd Manasa Potluri
Computer Science Department
George Mason University
Fairfax, VA, USA
mpotluri@gmu.edu

4th Jay Palan
Computer Science Department
George Mason University
Fairfax, VA, USA
jpalan@gmu.edu

I. Introduction

In a world where humans are actively relying on machines and intelligence derived from data, relying solely on a humans expertise is not efficient. According to the 2015 report released by Institute Of Medicine (IOM), most people will suffer from at least one wrong or delayed medical diagnosis during their lifetime. Americans experience about 12 million diagnostic errors a year [1]. This creates a gap in the market, a need for a system which can aid a healthcare practitioner to remember things he may forget, to provide information he may not have known, to prevent misdiagnosis, and to save lives. In our daily life, when we visit a doctor, the diagnosis he gives is based only on his education, experience and to some extent, intuition. Consider a case study provided in the IOM report [1] A 51-year old woman with a family history of heart disease repeatedly asked her doctor's' office to refer her to a cardiologist for a stress test. Three months after her initial request, on the day of her appointment, she died because of significant coronary artery disease.

One of the main reasons behind misdiagnosis is the healthcare professionals failure to link the symptoms given by the patient to the correct disease name. This is especially true in case of rare diseases. A practitioners exposure to patients from a variety of geographical regions, races and age-groups will decide the probability he will come across a rare disease, which is low for a majority of practitioners. This further increases chances of misdiagnosis for unfortunate patients with rare diseases or disorders. Years of training, education, and experience must give them enough knowledge to many rare diseases, but those diseases are easy to be forgotten in the real world. It is just the same as we do not recall every single textbook we read.

To overcome the above issue, we propose a Knowledge-based Disease Recommender System. The recommender system mainly consists of three parts: (1) A user-interface, (2) A SPARQL query endpoint and (3) The ontology holding the knowledge about diseases and their related symptoms. The user interface will take input from the user which will be either a list of symptoms, or a disease. The middleware will take these inputs, use advanced reliable query technique to query the ontology. A relevant output i.e. names of diseases or symptoms will be generated from the available data. During the patients visit, the system will prompt the all possible diseases to the healthcare professionals and let them to filter out the incorrect one. This approach can greatly reduce the possibility that a rare disease is ignored.

The proposed system runs on the server with the knowledge distributed over the network in the form of ontologies. We believe using ontologies as a design for the semantic web can have its potential in the future as it becomes popular among researchers. Moreover, such design offers flexibility [2].It enables different websites to implement and generate recommendations. Also, this design is promising as it will be well maintained and easy to modify.

The organization of the paper is as follows. In Section II, we talk about what are the market gaps we aim to fill with this project, and the questions we aim to answer. In Section III, we talk about research we did in the field of health data, health systems, recommendation systems and standardized medical terminology. In Section IV, we discuss our prototype system, a high level-design, and conceptualizing real world scenarios to the project architecture. In Section V, we discuss the steps we took to build the prototype, our front-end code, and building the ontology on Protege. As last, we concluded the paper in Section VII.

II. Problem Statement

The purpose of this project and this paper is to shed some light and provide some implementable solutions in the following areas:

- A system which can provide disease if provided a list of symptoms
- A system which can provide a list of symptoms if provided a disease
- An interface which can be used by a non-technical user who does not understand the concept of querying datasets
- A dataset which can give you the ICD-10 code of a disease
- A processed output which has some intelligence to it, including probability of having a disease.

In addition to the complexity of todays healthcare, medical practitioners face a number of challenges while diagnosing

patients symptoms. One of the challenges is the need to improve the quality of care, reduce inappropriate medication and reduce errors. Another common problem is the incomplete understanding of medical terms for people who lack medical knowledge. With this project, we also aim to put some responsibility of healthcare in the patients hands.

III. Related Work

We begin by introducing the key terminologies and concepts which form a core part of this project.

A. Knowledge-Based System

Knowledge-based systems are capable of making decisions based on the knowledge residing in them, and can understand the context of the data that is being processed. They broadly consist of an interface engine, and a knowledge base [3]. There are several such systems being used by healthcare professionals. They are being served as diagnostic assistance and treatment guidance. Knowledge based systems proved to be very useful in the current market, despite the fact that there are still many challenges [4]. For example, if a patient's disease is rare, complex, or the healthcare professional is inexperience, or he is not specialized in this domain, the diagnosis will not have an accurate result because the knowledge based system can only serve as a diagnosis support purpose in that case or it will likely to provide incorrect diagnosis. Several knowledges based systems failed to resolve this issue. On the other hand, our system will include a very comprehensive disease name database, managing to map the input symptoms to all the possible solutions. We believe that a rare and complex disease may exceed the level of expertise of a knowledge based system. For instance, the discovery of a new disease will be excluded by the system. Therefore, we leave such disease to healthcare professionals with special expertise in the domain. Such design can reduce the risk of misdiagnosis.

Onconcin [5] was developed at Stanford University. It is one of the earliest recommender systems and it has been employed in the Stanford Oncology Clinic since 1981. It provides treatment advice to physicians administering experimental chemotherapy to cancer patients.

ATHENA [6] provides guidelines for hypertension using EON architecture for blood pressure control. It encourages blood pressure control and recommends different choice of drug therapy in relation to comorbid diseases. ATHENA has a modifiable knowledge system. It is designed to allow clinical experts to customize the knowledge base to incorporate the evolved the disease symptoms.

B. ICD-10 Disease Codes

International classification of diseases-version 10 (ICD-10) is a diagnosis coding system used in hospitals for data retrieval and billing purposes. Every code represents a disease, condition, symptom, or cause of death. However, from our point of view ICD-10 represents an ontology. This similar format helps while billing and for patient to claim insurance. [7]

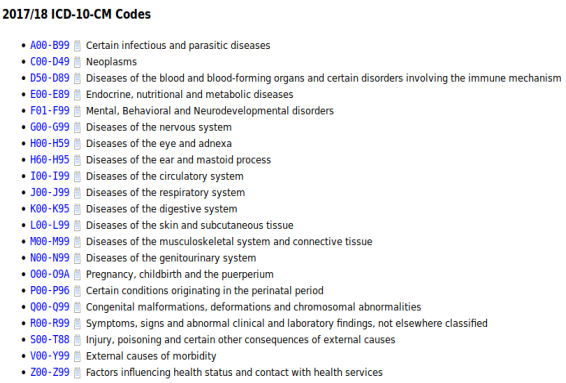

Fig. 1. Example of ICD10

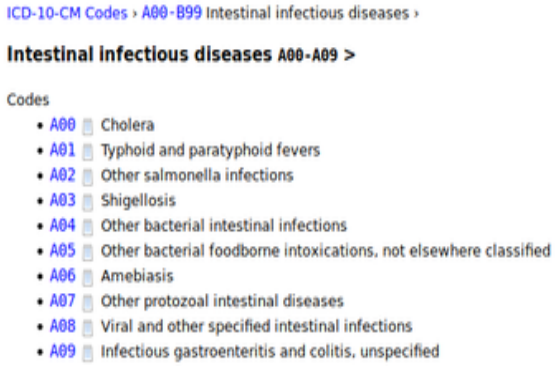

Fig. 2. Example of ICD10

Need for disease codes: These codes are important for patients as it makes it easier for them to take the responsibility of their healthcare in their own hands. Additionally, insurance companies use these codes for planning and deciding whether or not, and how much if yes, coverage they will provide. Figure 1 and Figure 2 are two examples for the ICD10 codes.

C. Web Ontology Language

The OWL, Web Ontology Language is a language for defining and instantiating Web Ontologies. In OWL ontology, concepts are arranged in hierarchical format with each concept represented by a node in the hierarchy. An OWL class having various properties and relationships with the other classes represents each node.

OWL is designed to represent rich and complex knowledge about things, groups of things, and relations between things. It has the capability to distribute across many systems. It is scalable to web needs and it is also compatible with web standards across the world. It is also open and extensible.

IV. Technical Approach

A. Use Case Development

The first step towards materializing any abstract research concept is mapping your requirements into a use-case diagram.

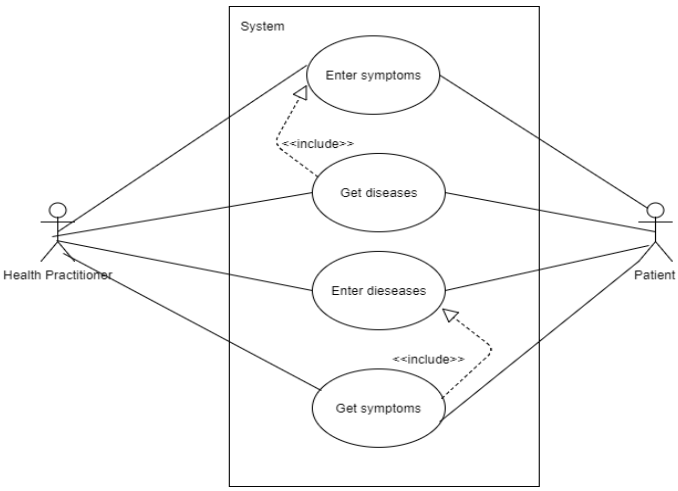

Fig. 3. Use Case Diagram

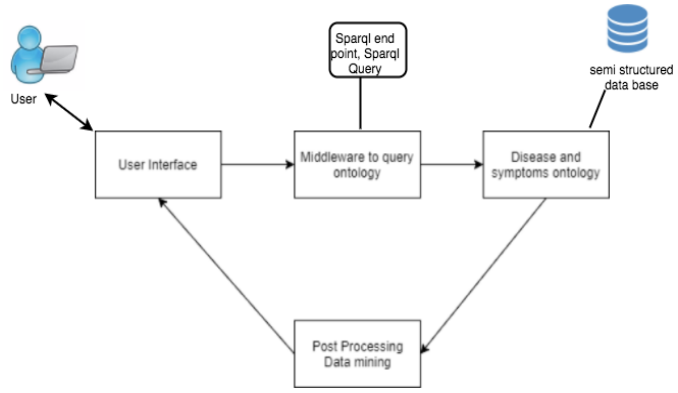

Fig. 4. Block Diagram

Having an outsider view to how a system is intended to interact with its users can later help to move inside into more details. Find below the use case diagram with two actors: Health practitioner and User. The rectangular box represents our system which is a black box at this stage, and the ovals represents what the actors should be able to do with our system.

Figure 3 sets some ground rules for the project to proceed:

- An actor/user should be able to interact in 4 ways with the system: Enter symptoms, Get diseases, Enter diseases, Get symptoms.
- Our target users are health practitioners and patients, whom we assume to be non-technical people.
- A directed relationship exists between pairs of use cases signified by <<include>>which means, Enter symptoms is a part of and needs to happen before Get diseases, and Enter diseases is a part if and needs to happen before Get symptoms.

B. Block Diagram

Figure 4 has four main blocks:

- The User interface will take inputs from the actors to get disease or symptom names
- The middleware is the SPARQL query which will query the ontology
- The ontology which holds the data, which has the mapping between diseases and symptoms.
- Post Processing where accuracy of the results is calculated

C. Knowledge Representation

When a system needs to exploit additional knowledge to generate recommendations, a knowledge-based approach is needed, as shown in Figure 5. Such knowledge may be elicited interactively by interacting with the user. Sometimes there is confusion in differentiating between content- and knowledge-based systems. Content-based systems typically work on text documents or other items, for which features can be automatically extracted, and for which some learning method is applied. In contrast, knowledge-based systems rely mostly on externally provided information about items.

Knowledge-based approach is advantageous, in scenarios, where collaborative filtering or content-based approaches show limitations. For example, buying a house, a car, or a computer is not a frequently made decision. Pure CF system would not perform well, because of the low number of available ratings. In our Scenario, the symptoms are not frequent. Every disease might not have similar symptoms. So, it is not feasible to have content based system. The most common division of knowledge-based systems is that between constraint-based systems and case-based systems. The recommendation process in both types looks similar: the requirements are elicited from the user and then solution is identified by the system. If the solution is not found, however, the requirements must be changed by the user. Optionally, explanation for the recommendation may also be provided to the user.

Detailed knowledge about items is pivotal for knowledge-based systems. Based on item feature values and requirements defined by the user, which are expressed as desired values or value ranges for an item feature, selecting the right item is the task of recommendation problem. It is also important to rank recommended items according to their utility for the user. Each item is evaluated according to a predefined set of dimensions that provide an aggregated view on the basic item properties.

Limitations:

- Patients accounts and understandings of what healthcare professionals did and said are, of course, their own perceptions. Accounts of the patient experience are of high importance in the current patient-centered provision of healthcare. Implying that result is given based on the experience response the patient gives.
- Data presented are based on patients retrospective accounts of their experiences, which may change with time. In addition, some patients accounts of diagnosis and interactions with healthcare professionals in the past years may reflect issues or ideas that are no longer prevalent in current practice.

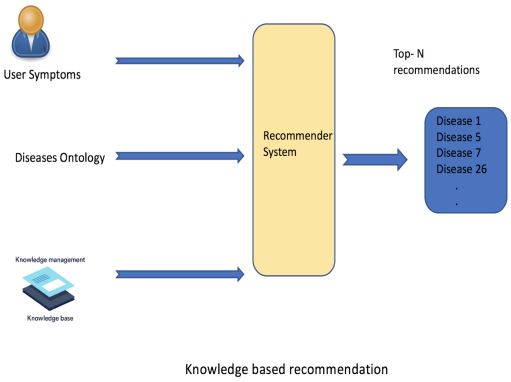

Fig. 5. Knowledge based recommendation system

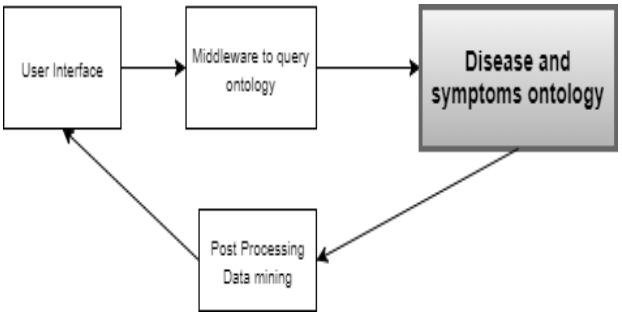

Fig. 6. Disease symptoms ontology

- Limiting to only few common diseases.

V. Implementation and Results

We have developed a recommendation system that is designed to support large-scale Web-based applications requiring highly accurate recommendations in real-time. This is done by drawing on patients symptoms and filtering the set of possible options to a more manageable subset. The proposed recommender system runs on the server with the knowledge distributed over the network in the form of ontologies. However, such recommender systems ignore the social elements of decision-making and advice seeking, and hence this model used by a medical practitioner who inputs the patients symptoms and knows possible diseases with maximum symptoms match. Most knowledge based systems on the market are found only to be effective on a narrow range of medical domain [4]. Such knowledge based systems do not contain in-depth knowledge like a doctor may have.

A. The Disease and Symptoms Ontology

As shown in Figure 6, this section introduces disease and symptoms ontology.

The following screenshots (Figure 7 8 9 10 11) show the Owl Editor Protege software. Figure 7 shows the Disease class with ICD10 codes, and the Symptoms class with all the available symptoms listed. The Figure 8 and Figure 9 show

Fig. 7. Disease Class with ICD10 Codes

Fig. 8. Instances Symptoms

the instances of symptoms and Object Properties. Figure 10 and Figure 11 provide some sample query results.

B. The User Interface

As shown in the Figure 12, User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions.

Everything stems from knowing your users, including understanding their goals, skills, preferences, and tendencies. Once you know about your user, make sure to consider the following when designing your interface:

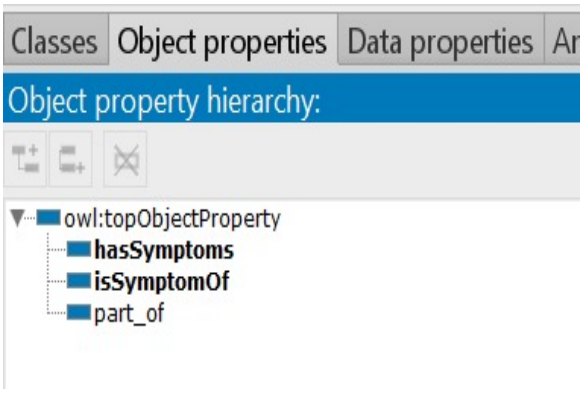

Fig. 9. Instances Object Properties

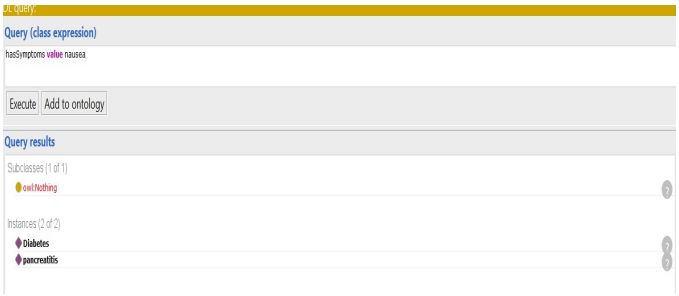

Fig. 10. Sample Query Result

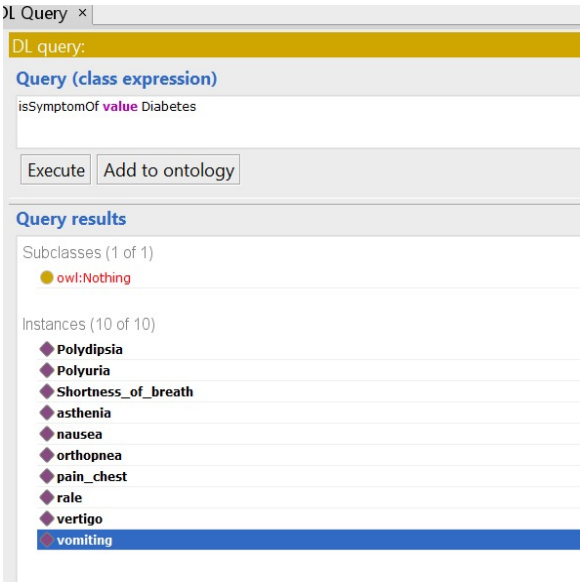

Fig. 11. Sample Query Result

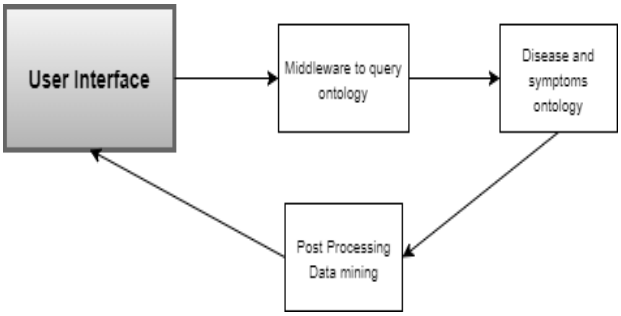

Fig. 12. The User Interface

- **Keep the interface simple.** The best interfaces are almost invisible to the user. They avoid unnecessary elements and are clear in the language they use on labels and in messaging.
- **Create consistency and use common UI elements.** By using common elements in your UI, users feel more comfortable and are able to get things done more quickly. It is also important to create patterns in language, layout and design throughout the site to help facilitate efficiency. Once a user learns how to do something, they should be able to transfer that skill to other parts of the site.
- **Be purposeful in page layout.** Consider the spatial relationships between items on the page and structure the page based on importance. Careful placement of items can help draw attention to the most important pieces of information and can aid scanning and readability.
- **Strategically use color and texture.** You can direct attention toward or redirect attention away from items using color, light, contrast, and texture to your advantage [9].

Make sure that the system communicates whats happening. Always inform your users of location, actions, changes in state, or errors. The use of various UI elements to communicate status and, if necessary, next steps can reduce frustration for your user. In our Disease recommender system, we kept the interface simple as it is used by non-technical people (medical practitioners). There are two major functionalities: one is find the diseases through given symptoms and other is find the symptoms of user given disease.

Figure 13 shows the Home Page of our system.

To find diseases from given symptoms: The user will input the major and minor symptoms from the drop down. They will also be given option to add more symptoms and can also enter symptom in text box if they dont find the symptom in the list, as depicted in Figure .

Figure 15 describes the output of our system. Our system will group the resulting disease into two sections, must have and may have. In must have section, we have high confidence that such disease can be correctly correlated to the symptoms. On the other hand, in may have section, we solely provide the possible results of the disease that the patient may have and such results are only served as an recommendation purpose.

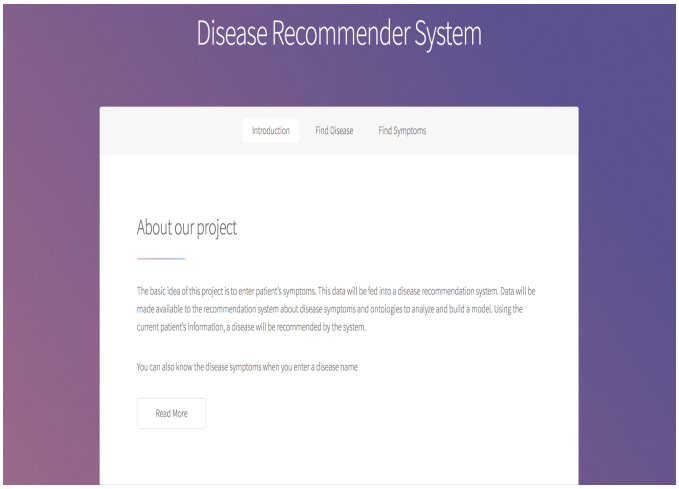

Fig. 13. Home Page

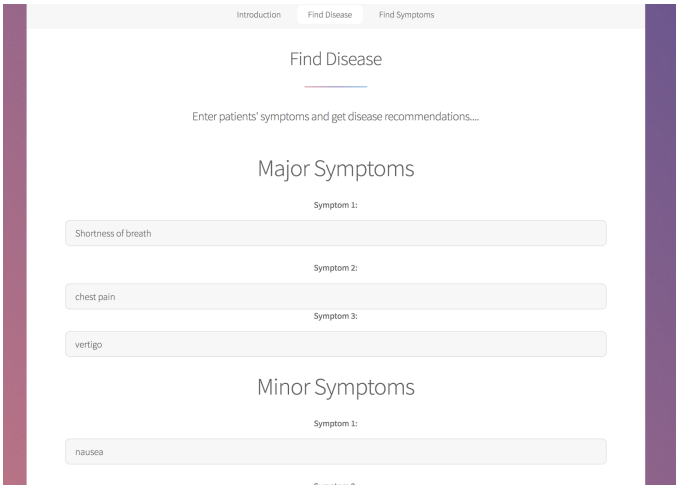

Fig. 14. More on Home Page

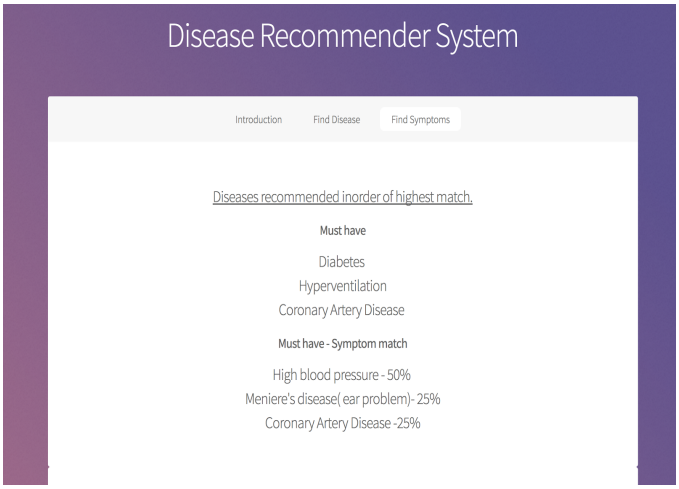

Fig. 15. System Output

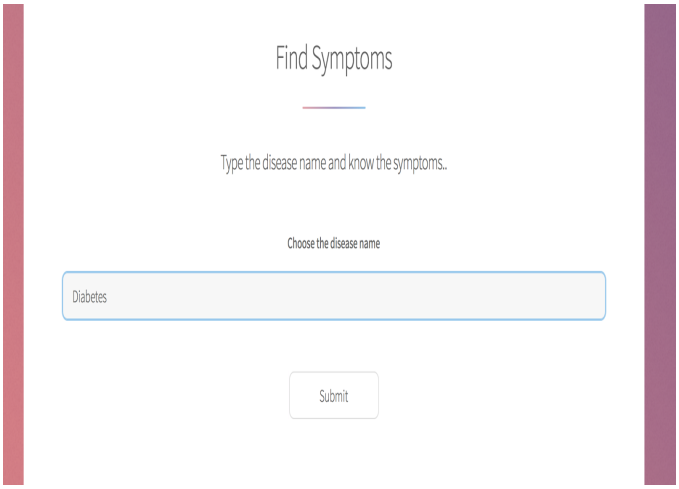

Fig. 16. Symptoms of Given Disease

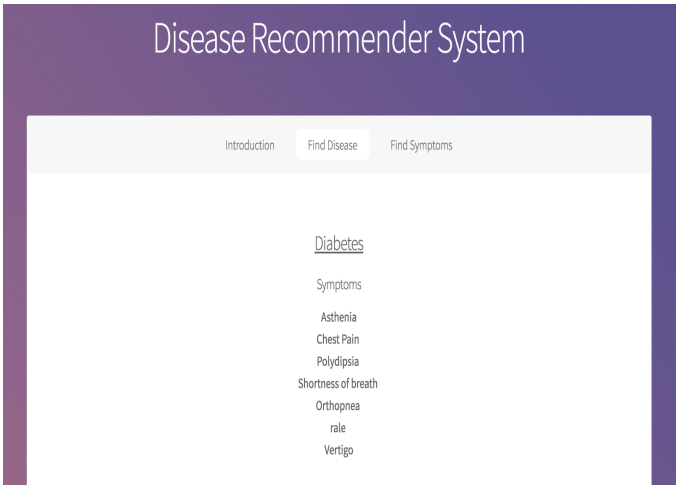

Fig. 17. Output of Disease Name

We leave the final decisions to doctors.

Symptoms of given disease: As shown in the Figure 16 and Figure 17, user can also get the symptoms by entering the disease by selecting from drop down.

C. Middleware to Query Ontology

This section will introduce the middleware to query ontology, as depicted by Figure 18.

What connects the User Interface to the ontology is the middleware. This middleware consists of SPARQL queries and a SPARQL endpoint where these queries can be executed. For the purposes of demonstration, we ran those queries on Protg software.

Figure 19 and Figure 20 show a sparql query and its results. We query the ontology to find the symptoms of disease Diabetes. The next image shows a similar query - names of diseases which have one of their symptoms as nausea

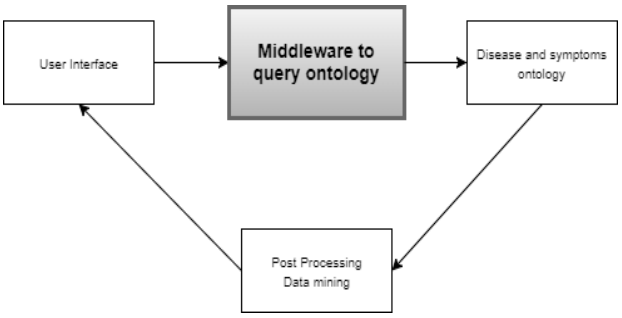

Fig. 18. Middleware to Query Ontology

```
SPARQL query:

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX xsd: <http://www.owl-ontologies.com/unnamed.owl#>
PREFIX xsd: <http://purl.obolibrary.org/obo/>
PREFIX EHR <http://www.jaypalan.com/EHR.owl>

SELECT ?Symptoms

WHERE {
EHR:Diabetes EHR:hasSymptoms ?Symptoms .
}
```

Fig. 19. Example of SPARQL Query

```
SPARQL query:

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX xsd: <http://www.owl-ontologies.com/unnamed.owl#>
PREFIX xsd: <http://purl.obolibrary.org/obo/>
PREFIX EHR <http://www.jaypalan.com/EHR.owl>

SELECT ?Disease

WHERE {
EHR:nausea EHR:isSymptomOf ?Disease .
}
```

Fig. 20. Example of SPARQL Query

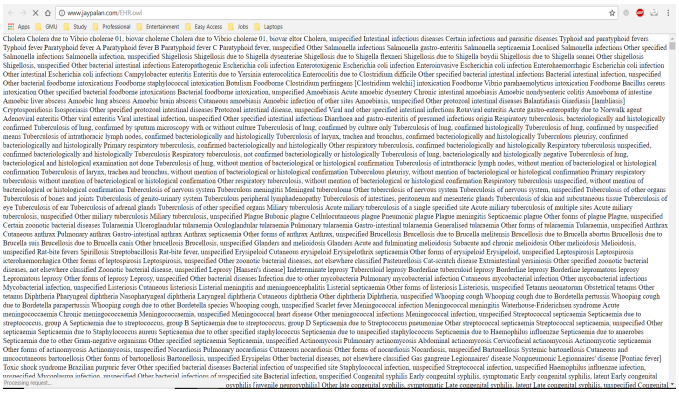

Fig. 21. Broken URI

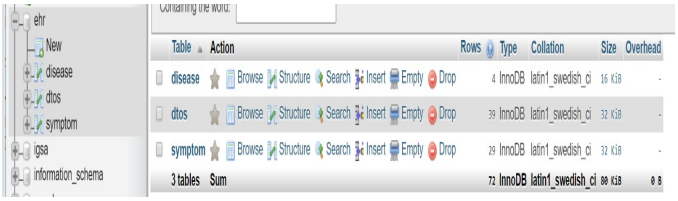

Fig. 22. MySQL Database Layout

D. Challenges, Problems and Solutions

For the UI to automatically run queries in the backend depending upon the user input, we need two things in place: (1) Host the Disease-symptom Ontology developed by us on the generate. (2) Generate a SPARQL-endpoint, which can convert user input into SPARQL queries to query the ontology.

We could successfully host our ontology at the link provided below.

http://www.jaypalan.com/EHR.owl

Attached Figure 21 is an image of how the ontology looks like.

What we could not achieve is creating a SPARQL endpoint to query this ontology. This was because the ICD-10 ontology which we used from http://www.owl-ontologies.com/unnamed.owl# has some problems. The URI is broken, and we are not able to use that link anymore.

To solve this problem and create a working model of our system, we converted the ontology from a semi-structured format to a structured form and stored it into a MySQL database, so that we could connect that to our User Interface and easily query it.

Figure 22 shows the our MySQL database layout.

The following figures (Figure 23 24) are examples of SQL queries which our User Interface can generate to get results.

E. Post Processing and Data Mining

From the SPARQL query results, we will get two types of output, must have and may have. Under each category, the data is ordered randomly. However, different diseases in each category may be related to the symptoms differently. For

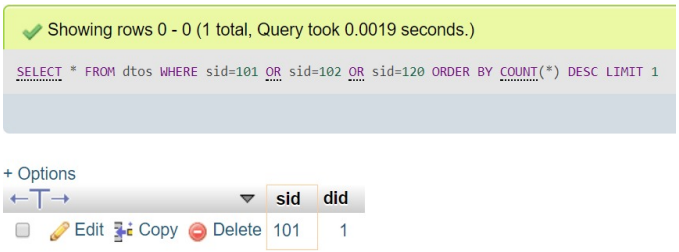

Fig. 23. Example of SQL Query

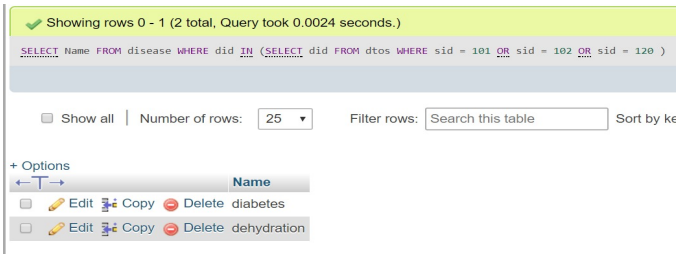

Fig. 24. Example of SQL Query

example, under must have category, disease 1 has symptom: fever and diarrhea. Disease 2 has symptom: fever, diarrhea, gas, vomiting, and dizziness. If the user queries fever and diarrhea, both disease will be returned. However, disease 1 is more related to the input symptom than disease 2. Thus, we will further sort the data, as shown in Figure 25.

1) Data Mining: To further sort the data, we will use data mining technique cosine similarity to calculate the similarity score. We first create a list S including all of the possible symptoms. Then we will create a list D including the returned diseases from the previous part. Next, we will construct a normalized symptom-disease matrix A using S and D. We will also construct query vector q using the query symptoms. After that, we use cosine similarity equation to compute the similarity score for each disease. We will provide an example later to better illustrate the methodology.

2) Cosine Similarity: The equation is shown in Figure 26.

In the equation, a_j is the column in the normalized symptom-disease matrix A and a_i is the row in the normalized

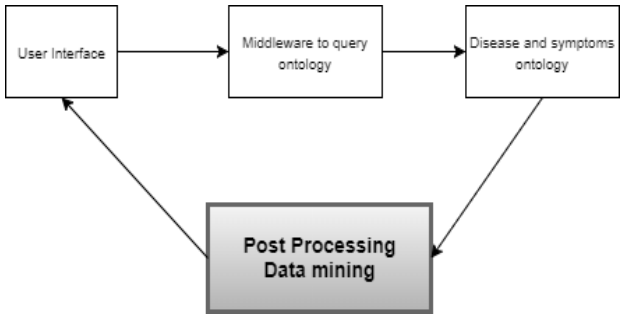

Fig. 25. Post Processing

$$cos\theta_j = \frac{\mathbf{a}_j^t \mathbf{q}}{\|\mathbf{a}_j\|_2 \|\mathbf{q}\|_2} = \frac{\sum_{i=1}^{9} a_{ij} q_i}{\sqrt{\sum_{i=1}^{9} a_{ij}^2} \sqrt{\sum_{i=1}^{9} q_i^2}}$$

Fig. 26. Cosine Similarity

matrix A.

For example, there are total five symptoms and three returned diseases, associated with the stored symptoms.

S(ymptoms):
S1: fever
S2: diarrhea
S3: gas
S4: vomiting
S5: dizziness
D(iseases):
D1: fever, diarrhea, gas
D2: fever, gas
D3: fever, gas, vomiting, dizziness

Then we will construct the symptom-disease matrix, which is a 5 by 3 matrix, as rows are symptoms and columns are diseases. If the disease has the symptom, we will assign the value to be 1, else it will be 0. The matrix is shown as follows.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

We then normalize the matrix so that each column has its Euclidean norm to be 1, as shown below.

$$\begin{bmatrix} 0.574 & 0.707 & 5 \\ 0.574 & 0 & 0 \\ 0.574 & 0.707 & 0.5 \\ 0 & 0 & 0.5 \\ 0 & 0 & 0.5 \end{bmatrix}$$

In the next step, we use the query symptom word to construct the query vector $q = (S1, S2, S3, S4, S5)$. If the word is in the query, the value is set to 1, otherwise it is 0. Assuming our query symptoms are: fever and gas. Our query vector will then be $q = (1, 0, 1, 0, 0)$. Now, we have the normalized matrix A and query vector q. We will use the following equation to compute the cosine similarity of each disease.

The computed results are:

$$cos(D1) = \frac{(0.574 \times 1 + 0.574 \times 1)}{\sqrt{(2)}} = \frac{1.148}{\sqrt{(2)}} = 0.812 \quad (1)$$

$$cos(D2) = \frac{(0.707 + 0.707)}{\sqrt{(2)}} = 1 \quad (2)$$

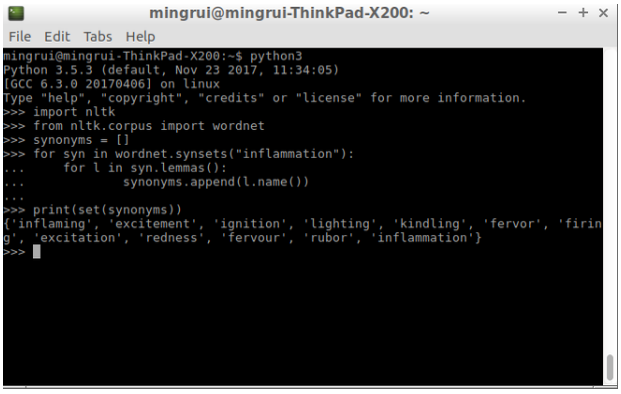

Fig. 27. Synonyms

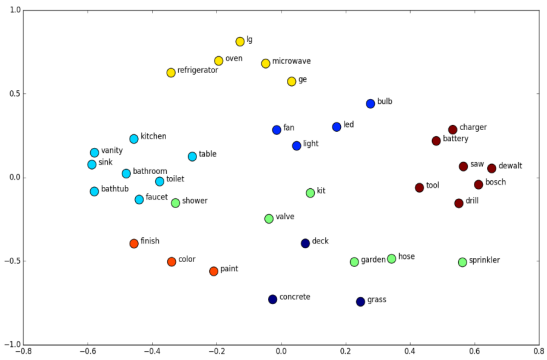

Fig. 28. Word Clustering

$$cos(D3) = \frac{(0.5 \times 1 + 0.5 \times 1)}{\sqrt{(2)}} = \frac{1}{\sqrt{(2)}} = 0.707 \quad (3)$$

The above results indicates that disease 2 has the highest similarity score and disease 3 has the lowest similarity score. As a result, we will sort the result as D2, D1, D3.

This post processing technique is applied to both must have and may have sections separately to provide a more accurate result.

VI. Future Scope

In the future, our ontology can be constantly updated with additions of new diseases and symptoms for them. It is also advisable to make our system modifiable to the healthcare professionals to that our system can be constantly updated with the accurate and the latest information.

Sparkle endpoint can be created as well, and UI can be directly connected to it.

In the future, our system can also let user input more parameters such as age, gender, disease history, which could potentially aids our system to reach a more accurate result.

Sometimes the patient who is suffering from the disease may not know the accurate medical term to describe his symptoms or just do not have the capability to use complex English words. In this case, our drop-down menu will not be suitable anymore.

To solve the issue, we will improve our application to be more open-ended. It will have the capability to accept any input words instead of just provided symptom words. For example, we will have word hemoptysis under drop down menu. Hemoptysis means coughing up blood mucus. However, the patient does not know the definition of hemoptysis. The new design will solve this by letting user enter symptoms in their own words. In this case, they can simply enter coughing up blood mucus

To query the ontology, we provide two approaches. First, we can apply synonym library on the input word and use the synonyms to query the data. There are several existing synonym libraries. We have an example for ”wordnet” as shown in the Figure 27.

However, the pure synonym approach will not be accurate enough as the meaning of word keeps changing as time goes by.

Another approach can be using the semantic meaning of a word, and query based on the meaning of the input word. To extract the semantic meaning, we can use some semantic embeddings: PCA, Word2Vec, Glove, and etc. Such semantic embedding tools adopt deep learning technique and they are more accurate than just synonyms. An illustration of the work is shown in Figure 28.

VII. Conclusion

In this work, we managed to create our knowledge-based disease recommender system. We have created the user interface for our recommender system and we constructed the ontology for the system. Then we implemented the middleware and used DLquery to retrieve result from our database. In the next step, we described the challenges and how we overcame them. Then we applied the data mining technique as our post processing on the result.

The medical field is vast and complicated and needs a lot of research and effort. Its difficult but not impossible to build such a system.

We wish that, in the future, this kind of system can be used to save lives.

References

[1] Misdiagnosis in America: Shocking Statistics, Mercola.com. [Online]. Available: https://articles.mercola.com/sites/articles/archive/2015/09/30/diagnostic-errors.aspx. [Accessed: 03-Dec-2017].

[2] P. Bedi, H. Kaur, and S. Marwaha, Trust Based Recommender System for the Semantic Web, in Proceedings of the 20th International Joint Conference on Artifical Intelligence, San Francisco, CA, USA, 2007, pp. 26772682.

[3] What is a Knowledge-Based System (KBS)? - Definition from Techopedia, Techopedia.com. [Online]. Available: https://www.techopedia.com/definition/7969/knowledge-based-system-kbs. [Accessed: 03-Dec-2017].

[4] E. Coiera, Guide to Health Informatics, 2Ed. CRC Press, 2003.

[5] G. M. Kahn, M. L. Fagan, and H. E. Shortliffe, Time in Clinical Decision Support Systems: Temporal Reasoning in ONCOCIN and ONYX, SIGBIO Newsl, vol. 8, no. 1, pp. 1316, Mar. 1986.

[6] M. K. Goldstein et al., Implementing clinical practice guidelines while taking account of changing evidence: ATHENA DSS, an easily modifiable decision-support system for managing hypertension in primary care, Proc. AMIA Symp., pp. 300304, 2000.
[7] M. Popescu and M. Khalilia, Improving disease prediction using ICD-9 ontological features, 2011, pp. 18051809.
[8] Recommender System for Diagnosis and Treatment of Tinnitus - PDF. [Online]. Available: http://docplayer.net/59629697-Recommender-system-for-diagnosis-and-treatment-of-tinnitus.html. [Accessed: 03-Dec-2017].
[9] User Interface Design Basics, 21-May-2014. [Online]. Available: /what-and-why/user-interface-design.html. [Accessed: 07-Dec-2017].