Smart interactive search for Vietnamese disease by using data mining-based ontology

Hong Son Nguyen\textsuperscript{a}, Minh Hieu Le\textsuperscript{b}, Chan Quan Loi Lam\textsuperscript{b} and Trong Hai Duong\textsuperscript{c}

\textsuperscript{a}Faculty of Information Technology, Le Quy Don University, Ha Noi, Vietnam; \textsuperscript{b}International University, Vietnam National University HCMC, Ho Chi Minh, Vietnam; \textsuperscript{c}Institute of Science and Technology of Industry 4.0, Nguyen Tat Thanh University, Ho Chi Minh, Vietnam

\textbf{ABSTRACT}

In disease information retrieval, usually, users only know a top of basic information about the disease such as \textit{headache} and \textit{high fever} symptoms. Most search engines rely on keyword search and do not return exact results because they only literally find the records matching with the keywords of the queries. In this work, we propose a novel interactive search for disease information by using data mining-based ontology. In particular, a human disease ontology is used as a knowledge base to semantically recognize the input for the search engine. The association rule is applied to generate associated relations among keywords (including symptoms, derives from, located in…) for interactive query refinement. A Bayesian-based ranking algorithm is also proposed to arrange the search result. Prospectively, our approach is valuable not only to increase the accuracy of searching human disease, but also provide a significant approach in data mining ontology.

\textbf{ARTICLE HISTORY}

Received 1 August 2016
Accepted 3 April 2017

\textbf{KEYWORDS}

Interactive search; faceted search; semantic search; ontology; data mining; association rule; Bayesian

\section*{Introduction}

The availability of information for treatment and healthcare on the Internet has helped people increasing knowledge about disease prevention and even treatment for themselves and others. As a consequence, research institutions, clinics and hospitals consecutively provide online medical knowledge. Then, users currently look up a disease hierarchically by index and read a considerable amount of information to get a desired reference. Due to a vast majority of unverified sources of information, Internet users, who do not have adequate expertise knowledge in medicine, tend to apply other familiar disease treatments onto their own cases, which possibly lead to mistreatment.

Most search engines rely on keyword search and do not return exact results because they only literally find the records matching with the keywords of the queries. For example, a user who does not know of ‘\textit{Hemorrhagic fever}’ but its symptoms such as \textit{headache} and \textit{high fever}. These keywords are then inputted as a query into the search box.

\textbf{CONTACT} Trong Hai Duong haiduongtrong@gmail.com

© 2017 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
Therefore, thousands of returned results matching with keywords including *high fever* and *headache* make it difficult to find the information about ‘Hemorrhagic fever’.

**Interactive search**

Google, Yahoo and Bing search engines are using semantic search technology to interpret the keyword meaning and predict the user intent. In some cases, they do not satisfy the user’s need due to their lack of information and a huge number of returned results. On the other hand, an interactive search engine can improve search result by supporting the users with questions and feedbacks (Ruthven, 2003). Depending on the answers during the interaction, the search scope can be refined to give the results which are more precise. When users start a search with initial query, the system uses a retrieval information algorithm to provide the users with questions, suggestions or word substitution related to the user’s query (Kelly, Gyllstrom, & Bailey, 2009). When users reply to this recommendation, the system will understand more detail about the user intent. This process can be repeated many times until the users get their desired results.

An interactive search engine should understand and translate natural language keywords to semantic query and execute it on a knowledge base (Bernstein & Kaufmann, 2006; Popescu, Etzioni, & Kautz, 2003). The translation should return the results in accordance with user’s intent. NaLIX system (Li, Yang, & Jagadish, 2005) uses an xml database and has a generic interactive natural language query interface. The input terms are parsed into XQuery expression and mapped to the proximity of corresponding elements in the xml database. NaLIX also helps a user to input the passable queries by providing meaningful feedback and helpful rephrasing suggestion.

SemSearch engine (Lei, Uren, & Motta, 2006) is different with NaLIX approach. It interprets the keywords by using instances, concepts or properties in the back-end ontologies. The system performs a text search to find matching records as an input and uses this combination to produce appropriate formal queries. The complex queries including multi keywords can be allowed in SemSearch engine, but it will create an explosion amount of semantic entity match combination.

Royo et al. uses another mechanism to translate keywords into semantic query (Royo, Mena, Bernard, & Illarramendi, 2005). The system uses a general-purpose third-party thesaurus, WordNet and a pool of pre-existing ontologies to discover different meanings of the user keywords. The disambiguator mobile agent of the system can recommend users with different meanings of keywords. Moreover, the knowledge engineer mobile agent can also obtain the relationship among the semantic keywords and terms in the context. Then, it finally returns a list of semantic queries and ranks them based on the synonymy probability.

In the medical domain, there are numerous data about diseases, which including animal models, human cell lines and tissue specimens are incessantly developed. Semantic search is required for this pool of rich information. Recently, ShahimEssaid (Shahim, Carlo, & Melissa, 2013) built a concept-based semantic search engine in a bio-specimen repository in order to help researchers find relevant bio-specimens or models of diseases. The data source of this search engine includes the department of Pathology and Knight Cancer Institute repository. Subsequently, the Unified Medical Language System (UMLS) developed by the NLM is used to reprocess data. It provides the MetaMap-
named entity recognition tool that is specifically trained for biomedical text. MetaMap tool parses the surgical pathology reports of the repository and produces a set of UMLS concepts. The queries will perform a free text search based on concepts identifiers and return relevant bio-specimens via the concept mapping location information. The limitation of building this search engine is that the non-natural language patterns of text in the repository might affect how to parse the text, identify noun phrases and map concepts to the text segments. Furthermore, it is also difficult to experiment with different mapping tools since it frequently requires a database schema modification. Another concept-based semantic search engine in medical domain is the research of Bevan Koopman (Koopman, Bruza, Sitbon, & Lawley, 2012). Its data source is composed of SNOMED-CT (see Note 2) ontology and a real-world collection of medical records. In order to extract concept, it utilizes MetaMap to transform the term-based originals of documents into UMLS concepts as defined by the SNOMED-CT ontology. In some cases, the UMLS concepts have no equivalent definition in SNOMED-CT.

Disease ontology

Scientists have studied and documented the variation of human health to unravel the mystery of human disease (Schriml et al., 2012). Data from multiple sources are unified and integrated into Disease ontology (DO) on semantic webs. DO facilitates the evaluation of diagnostic and treatment and comparison of data between studies based on semantic annotations of diseases. The initial version of DO was created in 2003 and 2004 by using ICD-9 as the foundation for DO vocabularies. Later versions of DO were improved and reorganized based on UMLS disease concepts including the term concepts of SNOMED-CT and ICD-9 (Schriml et al., 2012). Then it has become a community-driven, open and extensible framework using semantic relationships. DO are continuously being improved and extended with new DO terms. It currently has a single structure for disease classification and provides a clear definition for each. However, the relationship between disease causes and symptoms, as well as signs of diseases in this ontology, have not been systematically built yet. These relationships will be exploited to create the interactivity between the user and the disease in the new semantic search engine.

The ontology-based search engine can infer and provide users with more relevant results (Stojanovic, Studer, & Stojanovic, 2014). It interprets the keyword queries input by the user and translates them into semantic queries to access the ontology repository. A new search engine proposed in this paper would translate semantically search queries and interact with the users to capture their intent. Besides, the association rule (Neesha & Chander, 2014) is utilized to infer a list of questions to the user. Then, the answers iteratively help the search engine apprehend user intent and narrow down searching scope.

Data mining-based ontology for interactive search on disease domain

We proposed a new interactive search engine based on the DO. It has an ability to infer concepts from a query and suggest the inferred concepts to the user in order to make it interactive with them. Disease database is extracted from DO by using semantic query language. Association rules based on Apriori algorithm exploit disease data to
support interaction between users and the system. For arranging the search result, a Bayesian-based ranking algorithm is also provided. Figure 1 is the system architecture.

DO is a hierarchy of diseases. Each disease may have multiple levels of a subclass or superclass. A subclass or superclass is also a disease. And, a disease can have no subclass. In Figure 2, ‘Bệnh truyền nhiễm’ (disease by infectious agent) has multilevel of the subclass. ‘Bệnh nhiễm khuẩn do nấm’ (fungal infectious disease) is a subclass level of it or we can say that it is a superclass level of the fungal infectious disease. In addition, the category also has other subclass levels such as ‘Bệnh nấm da’ (cutaneous mycosis), ‘Bệnh nấm ngoài da’ (ermatophytosis). ‘Bệnh nấm da’ and ‘Bệnh nấm ngoài da’ are at the same level and also subclasses of ‘Bệnh truyền nhiễm’. Besides, ‘Bệnh nấm da chân tay’ (basidiobolomycosis) is a disease that has no subclass.

A disease has a label, definition, its subclass, its superclass and its properties. The disease property or disease factor includes symptom, cause and location. Figure 3 is a definition for ‘Bệnh nấm da chân’ (tinea pedis) and it is a subclass of ‘Bệnh nấm ngoài da’ (ermatophytosis). Its symptoms are including ‘nứt’ (spina), ‘có vảy’ (squamous) and ‘chết lớt, làm xói mòn giữa các ngón chân, và trên bề mặt gan bàn chân’ (maceration, and eroded areas between the toes and on the plantar surface of the foot). Also, its location is ‘chân, đặc biệt giữa các ngón chân’ and it has caused such as ‘Trichophyton’ and ‘Epidermophyton’.

**DO extraction**

The semantic search engine is a mechanism that queries on ontology repository, so it must use the syntax of semantic queries to access the ontology.
Users tend to search with natural language query rather than a semantic query for its complicated knowledge about ontology and repository schema (Reichert, Linckels, Meinel, & Engel, 2005). There have been recently several complicated mechanisms converting from natural language to semantic one (Cimiano, Haase, & Heizmann, 2007; Lopez, Pasin, & Motta, 2005). For example, QUICK search engine (Zenz, Enrico, Wolf, & Wolfgang, 2012) generates a complete semantic query space, that is, a set of all possible semantic queries for a given set of keywords. It first identifies all possible query patterns called query template, then generates semantic queries by binding keywords to query templates. 

SemSearch engine (Lei et al., 2006) uses another approach. For detail, instances, concepts and properties are determined from the keyword queries and then combined to produce appropriate formal queries. It is time-consuming if the natural language query has been complex since ‘SemSearch’ engine may create an explosion number of semantic entity match combinations. In order to solve this problem, we extract all definitions, properties and relationship from DO and pour them into MySQL database. It will be faster and more flexible for our semantic search engine to access the database with MySQL queries.

**Figure 2.** A hierarchy of diseases.

**Figure 3.** A definition for ‘Bệnh nam ngoài da’.
Jena framework is used to extract data from DO. Jena is a semantic web framework, it
provides a programmatic environment for RDF, RDFS and OWL which are data represen-
tation models for ontology. SPARQL language is utilized to retrieve and manipulate data
stored in RDF and OWL format. SPARQL allows a query to consist of triple patterns, con-
junctions, disjunctions and optional patterns. The process of disease factor extraction is
illustrated in Figure 4.

Firstly, all diseases and their subclasses or superclasses are extracted from DO and
inserted into table ‘DiseaseIndex’ and ‘DiseaseHierarchy’. The table ‘DiseaseIndex’ included
‘DiseaseId’ and ‘DiseaseLabel’. Then, table ‘DiseaseHierarchy’ includes ‘DiseaseId’ and ‘Super-
DiseaseId’ which is a superclass of ‘DiseaseId’ or we can say ‘DiseaseId’ is a subclass of
‘SuperDiseaseId’.

Secondly, all disease properties or disease factors including symptom, cause and
location are derived. In the DO, these disease factors are in the definition, so we must
create a table ‘DiseaseDefinition’ first and we extract its records from DO. Besides, the
table ‘DiseaseDefinition’ included is ‘DiseaseId’ and ‘DiseaseDef’ which is the definition of
the DiseaseId. In the disease definition, the label of symptom is ‘có_triệu_chứng’ (has symptom),
the label of cause is ‘vật_gây_bệnh’ and the label of location is ‘ưu_trụ_tại’ (located_in) (see Figure 3).
Subsequently, we extract all disease factor values from all disease definitions and insert
them into disease factor tables including table ‘Symptom’, table ‘Cause’ and table ‘Location’.
Likewise, the disease factor table contains ‘FactorId’ and ‘FactorValue’ which is the value of
‘FactorId’ (Figure 5).

Thirdly, the Id of each record in disease factor table and the Id of the disease containing
the disease factor of the record are inserted into table ‘Factor-Disease’. Thus, the table
‘Factor-Disease’ includes ‘DiseaseId’ and ‘FactorId’ (Figure 6).

Figure 4. The process of disease factor extraction.
Retrieval algorithm

User’s queries may contain one or more disease factors in order to search for the diseases matching with these factors. Retrieval algorithm for the queries requires combining multiple tables together to have enough fields for the matching. The system also creates the full text index for many fields of many tables to increase the searching speed. Search result ranking would be based on the support, the confidence of Association Rule algorithm and the probability of Bayesian algorithm.

Hint retrieval

This procedure is performed when users enter a query to search. Space key press event of the user will call a function to access the ‘DiseaseFactor’ tables to suggest other similar disease factors. This list of hints will be included in the drop-down list of the search box. Simultaneously, these hints are also stored in the cache for later retrievals.

```java
Dlist = getDisease();
For each D in Dlist
  Do
    Vlist = getFactorValuesFromDefinition(D.definition);
    For each V in Vlist
      Do
        FactorId = InsertFactor(V); // insert into table DiseaseFactor
        InsertFactor_Disease(FactorId, D.Id); // insert into table Factor-Disease
      End For
    End For
End For
```

Figure 5. Disease factor extraction algorithm.

Figure 6. Get factor values from definition algorithm.
**Disease retrieval**

Once users have completed the query and clicked the search button, the system will execute the query to disease database to retrieve the matching results. This query is performed on multiple tables joined together such as 'DiseaseIndex', 'Factor-Disease', 'Disease-Factor', 'DiseaseHierarchy' and 'DiseaseClass'. The results' probabilities of the query will then be calculated based on Bayesian algorithm. The probabilities will be sorted and displayed in descending order.

**Disease factor recommendation retrieval**

In addition to getting the diseases matching the query, the system also accesses to the 'DiseaseRule' table to obtain the disease factors related to the query and recommend them to the user. This 'DiseaseRule' table is generated earlier from exploiting the disease database by Apriori algorithm. Disease factor recommendation retrieval of the system is to get the rules in this table which have 'LeftSide' fields containing the user's query and recommend the 'RightSide' field to the user. However, the order of disease factors in 'LeftSide' field of 'DiseaseRule' table may be different with the order of disease factors in the user's query. Therefore, in order to match more easily the 'LeftSide' field with the query, we extract the 'LeftSide' field of this table and insert them into 'LeftRule' table. This 'LeftRule' table includes 'RuleId' which is the rule id of the 'LeftSide' and 'FactorValue' which is the disease factor in the 'LeftSide'. These rules are also calculated confidences and supports. The ranking of recommended disease factors bases on these confidences and these supports.

**Hint suggestion**

Hint suggestion would increase the interactivity between the users and the system. When users enter the keyword into the search box, the query autocomplete functionality of the system that will prompt the user with similar queries for choosing. This feature helps the user to select the query more precise and accuracy to their intent and that will make the results more relevant. It not only helps to fill out the query faster by selecting the query from hints but also gives spelling suggestions to users. When the users just remember the first letters of a keyword and enter them into the search box, the system will suggest the keywords that contain these letters and the users can choose the best suitable keyword from the suggestions. In another case, users just only remember a part of the query. So, query autocomplete will be very useful in this case. The queries of the hint which are similar to the user's query will help them to choose the query most efficient for them. In addition, the suggestion hints not only help users choose the correct queries for their intent, but also help them have the queries belonging to the dictionary of the system. This will provide the high confidence in the query and the system will return more accurately in the results.

Figure 7 shows the process of getting hint suggestions from the system back-end in order to return to the user. First, the user will enter the keyword into the search box. After each user's key press event, the system will take query in the search box and check the cache whether it contains this query or not. If so, the system will get a list of hint suggestions of this query from the cache and suggest them to users. For instance, the query is not in the cache, the system will query table 'DiseaseFactor' of the database
to select the records containing the query and save them with the query in the cache. Then, these records will be the hint suggestions returned to the users. Moreover, in each hint suggestion, the query will be highlighted in order to help users more easily recognize the hints which are closest to the query.

**Disease rule extraction**

The diseases in disease database contain diseases factors. Each disease factor can be a property of disease such as symptom, cause and location. In this database, we can identify the factors of a disease but the relationship between these factors is still unknown. This relationship is also important in the diagnosis and in finding the cause of disease. For example, the user with a very painful bump on his back does not know the cause of this symptom and he wants to find out it. Based on the relationship of this bump symptom, we can determine the reason of this symptom is caused by a spider or a special ant appearing nearby. Another example, the user who has a fever and headache, but he does not know exactly what kind of his illness is. The relationship of these two symptoms can ascertain other related symptoms of the user who may have and all these symptoms should help him more easily to identify his disease by this semantic search engine. In order to exploit the relationship or association rule between the disease factors, we rely on the frequent occurrence of the disease factors in the disease set.

The well-known approach for mining association rules of frequent item sets is Priori (Han & Kamber, 2006). This algorithm is often used in analysing market basket database. This kind of database stores transaction set containing many items. The items in a transaction are the products that user bought at the same time. In our context, the items are the disease factors and the transaction is the disease which contains several disease factors. Let $I = \{i_1, i_2, i_3 \ldots i_m\}$ be a set of items. Let $T$ be the transaction and $\subseteq I$, $D$ is the set of transactions. Set of items we call $A$ and $A \subseteq I$. A transaction $T$ contains $A$ if $\subseteq T$. The form $A \Rightarrow B$ is an association rule, where $\subseteq I$, $B \subseteq I$ and $A \cap B = \emptyset$. Rule $A \Rightarrow B$ has support $s$. The $s$ is the ratio between the number of transactions containing both $A$ and $B$ and the total transactions of $D$. Confidence $c$ of rule $A \Rightarrow B$ is calculated by the ratio between the number of transactions in $D$ containing $A \cup B$ and the number of transactions that contain $A$. The association rule $A \Rightarrow B$ must satisfy both minimum support threshold and minimum confidence threshold which are set in the system.

$$\text{Support}(A \Rightarrow B) = \frac{\sigma(A \cup B)}{\# \text{of Trans}},$$

$$\text{Confidence}(A \Rightarrow B) = \frac{\sigma(A \cup B)}{\sigma(A)}.$$
**Apriori algorithm**

The Apriori algorithm applying on DO generates a set of association rules (Figure 8). The rule $A \Rightarrow B$ in this set means that $A$ is the disease factors of the queries user enter into the search engine and $B$ is the inferred disease factors from $A$. The disease factors of $B$ are recommended to the user following his queries. Then, these association rules are stored in the table ‘DiseaseRule’ (Figure 9). This table includes ‘RuleId’, ‘LeftSide’ (the disease factors in $A$ are separated by ‘&&’ notation), ‘RightSide’ (the disease factor in $B$), Confidence (confidence of ‘RuleId’) and Support (support of ‘RuleId’).

**Bayesian ranking**

The user’s queries return the diseases matching the disease factors in the queries. However, the specificity degree of the disease factors in the queries for each retrieved disease (retrieved disease is the disease matching the queries) is different. The results will be ranked by this specificity degree. We calculate this specificity degree based on the probability of Bayesian algorithm (Figure 10). Bayesian probability is a quantity that is assigned to represent a state of knowledge (Jaynes, 1986). Bayesian probabilistic specifies some prior probability, which is then updated in the light of new, relevant data. The

---

**Figure 8.** Apriori algorithm.

```
C_k: Candidate item set of size k
L_k: frequent item set of size k
L_1 = {frequent items};
For (k=1; L_k != ∅; k++) do
    C_{k+1} = candidates generated from L_k;
    For each transaction t in database do
        increment the count of all candidates in C_{k+1} that are
        contained in t
    L_{k+1} = candidate in C_{k+1} with min_support
End For
Return U_kL_k;
```

---

**Figure 9.** DiseaseRule table.
prior probabilities are the probability of class and the probability of attribute within a given class. In our context, the super classes of retrieved diseases are the classes and the disease factors in the queries are the attributes. In the returned result of our semantic search engine, diseases will be grouped by their superclasses and the superclasses will be ranked based on the probabilities of them for disease set in the queries. Let \( NC_i \) be the number of diseases belonging to disease superclass \( C_i \). Let \( NS_i \) be the number of diseases containing disease factor \( S_i \) and belonging to disease superclass \( C_i \). \( NTS_i \) is the total number of disease factor \( S_i \). The total number of diseases is called as \( NT \). The Bayesian probability (Han & Kamber, 2006) of the disease factor \( S_i \) within given class \( C_i \) is calculated by the equation:

\[
P(S_i|C_i) = \frac{NS_i/NC_i \times NC_i/NT}{NTS_i/NT}.
\]
Probability is calculated for multi attributes within a given class, Bayesian algorithm assumes attributes have independent distributions and thereby it estimates as following (Han & Kamber, 2006):

\[
(X|C_i) = \prod_{k=1}^{n} P(X_k|C_i).
\]

\(X\) in the above formula is a set of attributes with \(n\) attributes. In our context, let \(S\) be a set of disease factors in the queries and \(S\) has \(n\) disease factors. The Bayesian probability of a set of disease factors within a given class \(C_i\) is calculated by the equation:

\[
P(S \mid C_i) = P(S_1|C_i) \times P(S_2|C_i) \times \cdots \times P(S_n|C_i).
\]

\(P(S \mid C_i)\) is the probability of the superclass of retrieved disease for the disease factors in the queries. In order to calculate the \(P(S \mid C_i)\) faster when receiving the retrieved diseases, we extract all the superclasses of all diseases, calculate the quantities of them in advance and store them in the table ‘DiseaseClass’. Table ‘DiseaseClass’ includes ‘SuperClassId’ which is not allowed duplication and ‘Quantity’ which is the quantity of the ‘SuperClassId’. The quantity of a superclass is the number of diseases belonging to this superclass. The Figure 11 is the process of extract all superclasses of all disease. Firstly, we get superclass for each disease relying on ‘DiseaseHierarchy’ table. Then we select distinctly these superclasses and count the quantities for them.

**Interface**

The interface of the interactive disease search includes three sections. The first section is the search box with autocomplete drop-down list (Figure 7). This drop-down list shows the list of disease factors similar to the user queries. The disease factor of this list would help the user to complete his queries faster or help him in the case he just remembers a part of his keyword, he can rely on this disease factor as a hint to obtain his keyword.

The second section presents a list of disease factor inferred from the user’s query. This list is a list of checkbox in order to allow user to choose multi disease factors suggested from the system (Figure 12).

The final section shows a list of diseases which is the result returned from the user’s query. Each disease also has its Bayesian probability which expresses the probability of the relationship between the disease and user’s query.

---

**Figure 11.** Super class quantities extraction.
Experiment

In order to evaluate the accuracy of interactive disease search, we calculate precision and recall for the disease factor suggestions which are returned by the system. We randomly select five diseases such as ‘Số xuất huyết Korean’ (Korean hemorrhagic fever), ‘Bệnh thủy đậu’ (chickenpox), ‘Quai bị’ (mumps), ‘Cúm gia cầm’ (avian influenza) and ‘Bệnh dại’ (rabies). Each disease is assumed to be the search intent of one user. Each user starts the search query with a disease factor related to the disease that he or she is looking for. The system will suggest other disease factors related to the query and return results with matching diseases and the query at the same time. When users choose disease factors suggested from the system and click search, the system will offer new suggestions and newly returned results. This users’ selection is considered as the second time that users feedback are put into the system (the first time is the query when users start searching). Each user will perform five times of such communications with the system. Precision and recall of the suggestions or recommendations are returned by the system in a communication which are calculated by the average of the precisions and recalls of five users. In addition, precision is measured based on the number of suggested disease factors which belong to the disease users are looking for. Data noise cannot be completely removed due to uncommon data structure. The noisy data has the same content with the disease factor of the query that will be counted as a recall. Figure 13 is a graph of

Figure 12. Disease factor suggestion section.

Figure 13. Precision and recall measurement of system recommendation.
precision and recall measurement for the disease factor suggestions returned by the system in five times of communications. Table 1 shows the number of retrieved disease in five times of communications.

According to the graph of Figure 13, we can see that from the first communication to the fifth communication, the precision tends to increase gradually (from 15% to 65%). This means that when the system interacts with the user many times, it will upward destabilize about the disease that the user wants to search. It helps the recommendation of the system more relevant to user intent. Similarly, in Table 1, the number of retrieved disease tends to decrease (from 84 to 3) when the user interacts with the system several times. This makes the search scope of the disease user is looking for being smaller. Then the user will be easier to identify the disease relevant with his/her intent.

In addition to the accuracy evaluation, we also measure the efficiency of this interactive disease search engine by comparing it with Google search engine. We perform searching for five diseases above (the queries are the symptoms of these diseases) on both Google search engine and this Interactive disease search engine. The result of the test is shown in Table 2.

According to the table, we observed that the Google search engine takes a longer time to return a result than the Interactive disease search engine takes (the cases of ‘Quai bị’ and ‘Thủy đậu’ disease). In cases of ‘Bệnh dại’, ‘Sốt xuất huyết Korean’ and ‘Cúm gia cầm’ disease, Google search engine cannot return the relevant diseases or it returns a huge amount (millions) of result and user has to take much time to refine the results in order to find out the relevant diseases. The test showed that Google search engine can solely find the diseases having specific features or recognizable features. The strange diseases or the diseases having several common features cannot be found by Google search engine because it does not have the inference capability to interact with the users and capture more detail about the disease they are looking for. After all, interactive disease search engine has these capabilities and it returns the relevant diseases of the test in a short time.

### Table 1. The number of retrieved diseases in each communication.

| Communication       | Number of retrieved diseases |
|---------------------|------------------------------|
| First communication | 84                           |
| Second communication| 20                           |
| Third communication | 4                            |
| Fourth communication| 4                            |
| Fifth communication | 3                            |

### Table 2. Time consuming of Interactive disease search and Google search.

| Disease                                           | Interactive disease search time consumed (milliseconds) | Google search time consumed (milliseconds) |
|---------------------------------------------------|--------------------------------------------------------|-------------------------------------------|
| Bệnh thủy đậu (chickenpox)                       | 24                                                     | 84                                        |
| Quai bí (mumps)                                   | 25                                                     | 168                                       |
| Cúm gia cầm (avian influenza)                    | 89                                                     | No relevant result                        |
| Bệnh dại (rabies)                                | 35                                                     | No relevant result                        |
| Sốt xuất huyết Korean (Korean hemorrhagic fever)  | 34                                                     | No relevant result                        |
Conclusion

In this work, a novel interactive search for disease information using data mining-based ontology is proposed. The search engine is able to understand user-input by using a vocabulary from disease ontology. The associated pieces of information (including symptoms, derives from, located in . . .) of diseases are used for interactively refining query. A Bayesian-based ranking algorithm is also proposed to arrange the search result. According to the experimental result, the proposed approach is a significant approach.

Notes

1. https://uts.nlm.nih.gov/.
2. International Health Terminology Standards Development Organization (IHTSDO). Systematized Nomenclature of Medicine – Clinical Terms (SNOMED CT) (http://www.ihtsdo.org).
3. http://disease-ontology.org/.
4. https://Jena.Apache.org/.
5. ‘XML and Web Services in the News’.xml.org. 6 October 2006. Retrieved January 17, 2007.
6. http://En.Wikipedia.Org/Wiki/Autocomplete.
7. http://En.Wikipedia.Org/Wiki/Bayesian_Probability.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by Ho Chi Minh City Department of Science and Technology with Under grant number: 177/2015/Đ-KHCN.

Notes on contributors

Doctoral Student Nguyen Hong Son is the Chief of Training Department at Hue University of Law, Vietnam and the President of this University Commitee. He received an M.Sc. degree at the Faculty of Information Technology, Hue University of Sciences, 2005 and received the Bachelor of Computer Science with a certificate for best student of this University, 1997. At present, he a Doctoral Student of Information Technology at Le Quy Don Technical University. He has published around six academic papers in many domestic and international journals as well as conferences.

Minh Hieu Le is currently is a senior engineer at ELCA Information Technology, Vietnam. He received a Master’s Degree in IT Management at International University - VNU HCMC. His main interest is data mining.

Lam Chan Quan Loi is a researcher in computer science. He received B.S from Ho Chi Minh University of Science in 2007. He also received a half full scholarship of Masters course from International University-VNU. In 2016, he received a Information Technology Management Master degree. He is interested in researching about data mining.

Trong Hai Duong is currently a director of the Institute of Science and Technology of Industry 4.0, Nguyen Tat Thanh University, Vietnam. He received M.Sc. and Ph.D. degrees at the School of Computer and Information Engineering, Inha University, 2012. He received the Bachelor of Science with a certificate for the most excellent student at research at Dept of Informatics of Hue University, Vietnam, 2004. He received the Golden Globe awards and Innovative Youth badge in 2016. He was one of 68 representative young scientists who had a meeting with the Prime Minister of
Vietnam. He was one of 37 students of all Vietnam universities who won scholarship certificate of Ministry of Postal Service of Vietnam for the most excellent student of information technology, 2003. He has experiences in Information Engineering, especially Artificial Intelligence. His research interest focuses on Semantics and Bigdata. He has more than 60 publications.

References

Bernstein, A., & Kaufmann, E. (2006). GINO – a guided input natural language ontology editor. Proceedings of the International Semantic Web Conference (ISWC), Athens, GA, USA, pp. 144–157.

Cimiano, P., Haase, P., & Heizmann, J. (2007). Porting natural language interfaces between domains – a case study with the ORAKEL system. Proceedings of the International Conference on Intelligent User Interfaces (IUI), Honolulu, HI, USA, pp. 180–189.

Han, J., & Kamber, M. (2006). Data mining: Concepts and techniques (2nd ed.). Amsterdam: Elsevier.

Jaynes, E. T. (1986). Bayesian methods: General background. In J. H. Justice (Ed.), Maximum-entropy and Bayesian methods in applied statistics (pp. 1–25). Cambridge: Cambridge Univ. Press.

Kelly, D., Gyllstrom, K., & Bailey, E. (2009). A comparison of query and term suggestion features for interactive searching. Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information retrieval (SIGIR ’09). ACM, New York, NY, USA, pp. 371–378. doi:10.1145/1571941.1572006

Koopman, B., Bruza, P., Sitbon, L., & Lawley, M. (2012). Towards semantic search and inference in electronic medical records: An approach using concept-based information retrieval. Australasian Medical Journal, 5, 482–488.

Lei, Y., Uren, V., & Motta, E. (2006). Sensesearch: A search engine for the semantic web. Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW), Podebrady, Czech Republic, pp. 238–245.

Li, Y., Yang, H., & Jagadish, H. (2005). NaLIX: An interactive natural language interface for querying xml. Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, Baltimore, MD, USA, ACM Press, pp. 900–902.

Lopez, V., Pasin, M., & Motta, E. (2005). Aqualog: An ontology-portable question answering system for the semantic web. Proceedings of the European Semantic Web Conference (ESWC), Heraklion, Crete, Greece, pp. 546–562.

Neesha, Sh., & Chander, K. V. (2014, March). Association rule mining: An overview. International Journal of Computer Science & Communication, 5(1), 10–15.

Popescu, A., Etzioni, O., & Kautz, H. (2003). Towards a theory of natural language interfaces to databases. Proceedings of the International Conference on Intelligent User Interfaces (IUI’03), Miami, FL, USA, pp. 149–157.

Reichert, M., Linckels, S., Meinel, C., & Engel, T. (2005). Student’s perception of a semantic search engine. IADIS CELDA, Porto, Portugal, pp. 139–147.

Royo, J., Mena, E., Bernard, J., & Illarramendi, A. (2005). Searching the web: From keywords to semantic queries. Proceedings of the Third International Conference on Information Technology and Applications (ICITA’05), Sydney, NSW, Australia, IEEE Computer Society (2005), pp. 244–249.

Ruthven (2003). Re-examining the potential effectiveness of interactive query expansion. Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR ’03). ACM, New York, NY, USA, pp. 213–220.

Schriml, L. M., Arze, C., Nadendla, S., Chang, Y. W., Mazaitis, M., Felix, V., … Kibbe, W. A. (2012, January). Disease Ontology: A backbone for disease semantic integration. 4th International Conference on Biomedical Ontology, ICBO 2013, Montreal, Canada.

Shahim, E., Carlo, T., & Melissa, H. (2013). Enabling semantic search in a bio-specimen repository. ICBO, 1060, 50–53.

Stojanovic, N., Studer, R., & Stojanovic, L. (2014). An approach for step-by-step query refinement in the ontology-based information retrieval. Proceedings of the IEEE/ACM Web Intelligence Conference (WI), Warsaw, Poland, pp. 120–137.

Zenz, G. Zhou, X., Enrico, M., Wolf, S., & Wolfgang, N. (2012). Interactive query construction for keyword search on the Semantic web. Berlin: Springer.