International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.3 (2014), pp.187-202
http://dx.doi.org/10.14257/ijmue.2014.9.3.18

Ontology-based Question Answering System Development for Case Study of Thai Cats

Chantana Chantrapornchai[1], Archara Sangchan[2], Atitaya Kantankul[2], Chidchanok Choksuchat[2] and Suchitra Adulkasem[2]*

*[1]Department of Computer Engineering, Faculty of Engineering
Kasetsart University, Thailand*
[2]Department of Computing, Faculty of Science, Silpakorn University, Thailand

*[1]fengcnc@ku.ac.th, *suchitraa@hotmail.com*

Abstract

Question answering system requires knowledge and the effective representation to simplify the development and speedup the query process. In this work we attempt to apply the ontology design and develop a question answering system for the sample domain knowledge for Thai cats. The ontology and its engine are used to facilitate the query. We present the whole development process for such a system. First, the ontology for the Thai cats was designed. The domain knowledge is focused at the characteristics of Thai cats, the cat diseases, species of Thai cats. The principles of natural language processing is applied to process the question and forming the answer of the question correspondingly. The algorithmic process for finding the keywords, selecting keywords, and mapping keywords to the query is used. To simplify the processing, we employ the question patterns that are suitable for our domain knowledge. The patterns and keywords can be expanded to cover more cases in the future.

Keywords: *Thai cat ontology; Question answering; Thai cat information; SPARQL*

1. Introduction

Question answering system requires knowledge in the computer science domain [1]. Particularly, its challenge involves mainly the natural language processing, and knowledge representation. The natural language processing part is the front end part which relates to lexical and syntax analysis depending on language forms. The backend part relates to the knowledge representation and acquisition. This is the engine for searching the answer to the questions.

Question answering system can be categorized into open domain questions and closed domain questions. The open domain question focuses on searching the answer in the knowledge sources such as from the WWW while the close domain question kind focuses on forming the knowledge from sources and modeling the knowledge which is to help finding the answer easily.

The front end part of question answering system deals with the tokenizing the question and finding proper keywords to help search for the answer. This part involves the complex natural language processing which is specific to languages. For Thai language, this is even more difficult since Thai language contains many vowels and they can be put in three levels with the alphabets to forms words. Certainly Thai dictionary exists but it does not cover all contemporary words especially with the specific word domain.

The backend part of the system involves the keyword matching to the knowledge base. The strategy depends on the knowledge representation. It can be as simply as keywords matching with/without indexing or the intelligent matching with inference engine.

In this work, we are interested in building the prototype question answering system for Thai cats domain. We describe the prototype building process and several issues related to these two parts. The challenge on the front end part relies on the Thai language syntax and is addressed by the open source Thai natural language processing tool with some add-on while the challenge on the backend part is addressed by the ontology representation.

1.1. Related Work

There are many previous works that exploit ontology in many applications. For example, Sridevi and Nagaveni perform the document clustering based on ontology [2]. The key points of using ontology are different meanings of a keyword. The ontology helps define the relations between keywords and meanings. The document indices are extracted based on ontology and the particle swarm clustering is then executed.

Thomas, Redmond, and Yoon developed a case of E-commerce based on the ontology [3]. The rules are implemented based on SWR with the ontology representation. The motivation of the work is the extensive information for a user for shopping a particular item on the web. The developed expert system guides the user for seeking and accomplishing his goal.

Developing ontology is a very important step. While much ontology exists such as ontology on the website (*e.g.,* http://protegewiki.stanford.edu/wiki/Protege_Ontology_ Library#OWL_ontologies), a new ontology is more and more since the domain knowledge are extensive. Tools to develop ontology are a great help. Kim and Storey presented a methodology to construct an ontology called Webonto, where the information source is the World Wide Web and the web tool may be integrated [4]. The demonstration shows the application results.

Many question answering systems uses the semantic engine. For example, START is the open domain question answering system developed by MIT (http://start.csail.mit.edu/). It is still online. The system answers in many areas such as questions about places, weather, etc. It is based on Omnibase which provides an access to the outside data from many sources and from many kinds [5]. The input questions are converted into a style of object-property-value. It operates with the online data sources on World Wide Web to find the answer.

Sucunuta, Particular, and Riofrio demonstrated the architecture of question answering systems [6]. It contains several modules: the first module performs a lexical analysis from the input question while the second module accesses the data for answering. The third module performs the answer extraction.

PiQASso is a question answering system which uses semantic features to extract relevant paragraphs [7]. The system contains paragraph search engine and document indexing based on IXE. The index of full documents are created where the boundary of the information are marked. Minipar (http://webdocs.cs.ualberta.ca/~lindek/ minipar.htm) is used to analyze the sentences in a dependency tree structure. WNSense (http://en.wikipedia.org/wiki/Word_sense) is used for synonyms and word classification as an interface to the Wordnet (http://wordnet.princeton.edu/). The keywords are extracted from the questions as well as their parts of speech and generating the paragraph query. The questions are categorized as "How", "What" questions, *etc.* Word relations are used to find the answer nodes accordingly. The query is generated and may be expanded to find more answers. The answers are ranked and returned back to the user.

QAST is another question answering system for Thai Wikipedia. It is based on the open-domain question [8] (http://www.lsi.upc.edu/~qast/2009/). Two kinds of data sources are RDF and document index by Lexiton (http://lexitron.nectec.or.th/2009_1/). SPARQL is used for semantic queries. If the search from the RDF is not successful, the document index search is used. The types of questions acceptable are about place, organization, quantity, date/time, person. The system architecture mainly contains data representation, question processor, and answer processor. First, the SPARQL query is generated. There are two types: The question asking for definition and the question asking for property. In the answer processor, there is a ranking module which rates the answer. Then, the answer generator presents the answers in the sentential form.

2. Backgrounds

Ontology is the concepts of interested domains [4]. It describes the things in the domains and relationship among them. These relations are expressed in such a way that the semantics are represented and the semantic properties and axioms are exploited to \conclude the knowledge in that domain.

Typical question answering systems used natural language processing. The system needs the knowledge base to explore the answers. The sources of documents of knowledge may be from the local data, wikipedia, world wide web pages, etc.(http://en.wikipedia.org/wiki/Question_answering) Two types of questions are commonly used. First is the closed domain question where it is much easier to find answers in a local repository. In this manner, the NLP is usually used together with the ontology in a declarative style to facilitate the inference process. In the open domain question, they may depend on the world knowledge or general ontologies such as the work START by Info Lab Group at MIT (http://start.csail.mit.edu/). The source of data is from the World Wide Web.

Several steps are necessary to process question answering system [9-10] as inspired by TREC [7]. Starting from the question itself, the question classification is needed. For instance, we need to identify which kinds of questions we are accepting. It is a WH questions or yes-noquestion. If WH-questions are interested, what specifically it may be. One may be interested in What, When, Why questions, etc. With TREC, there are taxonomies of questions.

The same question may be asked in different forms with different keywords. That is the difficult part when dealing in a specific language. For example, one may ask what is the Nilachak's typical disease? In another way, one may ask, if the Nilachak cat has a red spot symptom, what is he likely to be? The answer may be the same but the questions may be different. In particular, the focus of the question should be identified as well as question keywords. The whole part here is called question processing. It is related to specific languages especially for the lexical and syntax analysis.

In the next step, it is concerned on the data source of the answer. It concerns about the representation of the data source. If it is an online repository, the crawler engines are needed. If it is a local repository, the data are needed to be stored. In any way, the representation of data is needed. Using the semantic query, the data may be extracted and represented in the RDF forms while with the paragraph documents may need indexing schemes to help searching to related contents. The important part here is to transform the question keywords into the query form. If the document indices are used, the question keywords need to be matched with the document paragraph algorithmically. For the semantic search, the keywords and parts of speech are needed to form proper queries.

The search engine may need to use the artificial intelligent techniques to help finding or concluding the answer properly. Without the inference engine tool, the answer needs to extract from the knowledge in the straightforward way only. The inference engine which goes with the knowledge representation will help the designer focus on the knowledge representation design and the programmer focuses on the mapping from questions to declarative query. The engine will do the rest. This is where the semantic search engine becomes important roles nowadays.

After the answers are retrieved, the answers are sorted or ranked. Then, answered sentences are needed to be formed. Again, the NLP becomes an important role. This part is sophisticated since it relies on the syntax or the language structure.

3. Methodology

To design the system, there are many steps to go through. It starts from the data sourcing and representation. Then the data properties are designed. Next, the template for questions are selected and their representation is designed related to the ontology concepts. After that, the front end part from the user interface and the controller engine is designed. The front end user interface takes inputs from the user question. Then, the lexical analyzer will analyze the possible related keywords. These are matched with the question templates to be translated to the related SPARQL query. After that, the answers are formed into a sentence and displayed.

3.1 Data Collection and Representation

We need to collect cat data from many sources. The selected sources [11-14] are used (http://www.ntsfarm.com;http://www.catthailand.com). These data are formulated are concepts as related ontology. Since there are many characteristics and aspects that should be considered. Figure 1 summarized our interested domain. The ontology is derived and adapted from many sources which described the ontology about animals and the ontology about diseases (http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_1.pdf) [15].

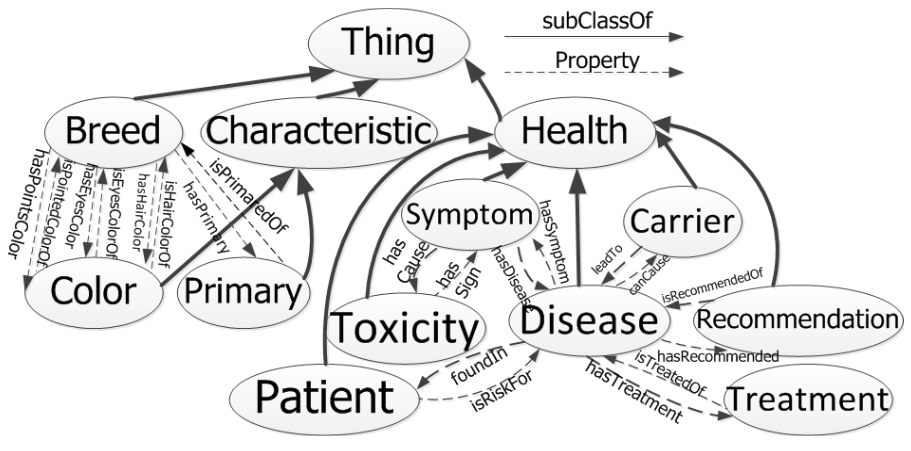


Figure 1. Thai cat ontology sample for query

In Figure 1, we summarize the classes and their relations. The knowledge we are interested here is about the cat characteristics, primary characteristics, breed, color. Also, the health concept considered is about disease, symptom carrier, toxicity, *etc*. Then, the sample properties are designed for these concepts as in Table 1.

Table 1. Property for ontology in Figure 1

Domain	Range	Property	Inverse Property
Breed	Color	hasHairColor	isHairColorOf
Breed	Color	hasEyesColor	isEyesColorOf
Breed	Color	hasPointsColor	isPointsColorOf
Breed	Primary	hasPrimary	isPrimaried
Disease	Symptom	hasSymptom	hasDisease
Disease	Recommendation	hasRecommended	isReccommendedOf
Disease	Treatment	hasTreatment	isTreatedOf
Disease	Patient	foundIn	isRiskFor
Symptom	Toxicity	hasCause	hasSign
Disease	Carrier	canCause	leadTo

3.2 Keyword Extraction and Preparation

After the ontology is designed, we obtain the owl file. Classes and properties in it needs to be related to the vocabulary for querying. From the above ontology in Figure 1, we categorize the vocabulary keywords and write them to text files.

Figure 2 shows the text files we create according to the ontology. The text files are generated based on the owl file which will find the vocabularies in each category. The vocabulary files are two types. That is the files for classes and the file for relations.

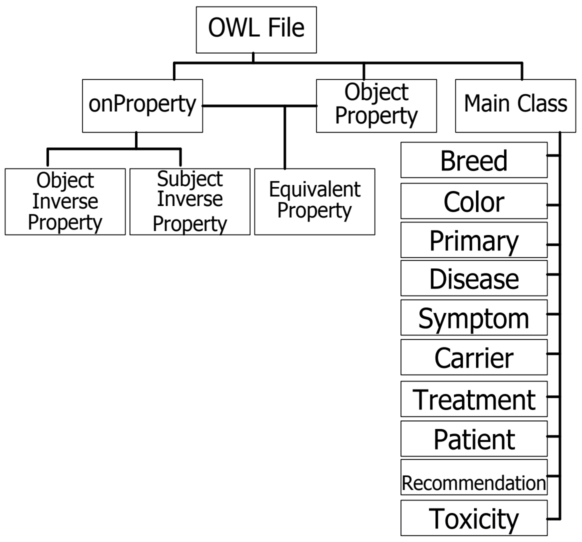


Figure 2. Text file generated related to the ontology

In details from Figure 2, there are the following relations.

1. Relation keyword I. It is the relation that is the object of the inverse relation.

2. Relation keyword II. It is the relation of classes in ontology or the subject of the inverse relation.

3. Relation keyword II. It is the all relations in the ontology. This is for the purpose of checking EquivalentProperty.

4 Breed

5. Color

6. Disease

7. Symptom

8. Treatment

9. Patient

10. Recommendation

11. Carrier

For the first two relations, we correspond it to the query form.

```
SELECT ? subject ? object
WHERE { ? subject owl:inverseOf ? object }
```

Particularly, the relation type I is the file with "object inverse property" and the relation type II is the file with "subject inverse property". For relation type III, it forms the query.

```
SELECT
WHERE { ? s rdf:type owl:ObjectProperty }
```

which is the other relation that has equivalent property. We take the results form this last query subtracted by the results from relation I and relation II to obtain the equivalent property.

For the class, we find the main ontology node from,

```
SELECT distinct ? cls.
WHERE { ? cls rdf:type owl:Class .
OPTIONAL {? cls rdfs:subClassOf ? superClass}.
FILTER ( ! bound( ? superClass)) }
```

That is we find the classes and subclassses of OWL:Thing and write them to files.

3.3 Front End and Query Processing

From the user input question in Thai, the question is tokenized using Lexto (http://lexitron.nectec.or.th/2009_1/) checking against our modified dictionary. Then from each word, it is compared with the vocabulary text file. In particular, the class and relation text files are checked. What we need is the name entity that specifies the classes and relations used to form a query. The query specifically needs class name, type of classes, relations, types of relations.

Then, we use them to form the query. We have the template for the query. The template consists of the variables about class name, class type, relation name, relation type. Next, from the relation, we find the domain and range. We check whether the class type matches with the domain or range of the relation.

If the class type is a domain of the relation, we have to find the inverse of the relation. If the class type is a range of the relation, it is used to query directly. In our prototype, we handle the following cases which may be further extended.

1. Class type is the range.

2. Class type is the domain. We check the relation type I or type II.

3. Class type is neither domain nor range/

4. Error case. There exists no relation, found only class name, or there exists no class name, found only relation/

5. These exists the keyword that shows the similar meaning such as similar to.

Note that the case 4 is for error case, and case 5 is the extension where we may have keywords identifying the equivalence where we did not define in the property. This may happen for the case like: "Which cat breed has the eye color the same as Nilachak"? Or we can say "Nilachak has the eye color similar to which breed?" This is the case where we do not consider the synonym of the vocabulary yet.

3.4 Answer Formation

After the query results are achieved, the answers are used to form a response sentence. The answer forming depends on cases as following:

1. If the question relation is the query relation, then the form is "answer + relation + class".

2. If the question relation is not the same as the query relation, the form is "class + query relation + answer".

In Figure 3, the overall architecture is presented. The architecture consists of many components. We divide them into levels.

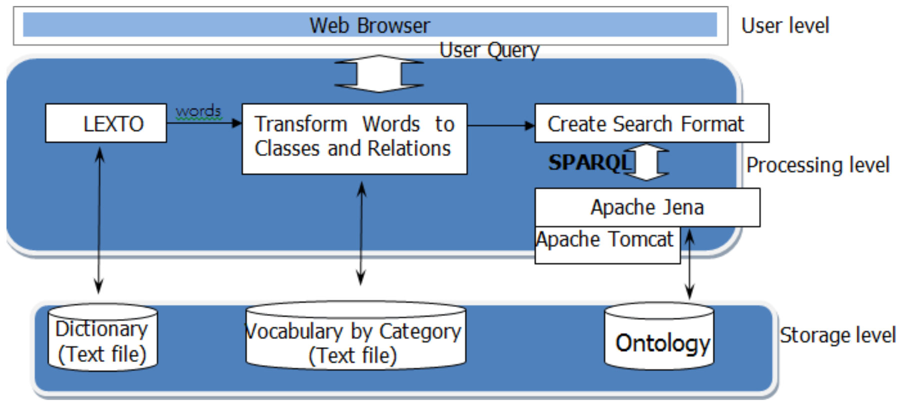


Figure 3. Overall system architecture

1. Storage level. In this level, we need dictionary which contains related words. This is obtained by Lexiton. It is saved as a text file. Also, the words related to ontology are kept.

Besides, we need the text file for vocabulary by categories. The categories are related to the ontology. The application is used to convert class in the ontology into these files. At last, the cat ontology is put in an owl file.

2. Processing level. There are components such as Lexto API which is used to tokenize Thai words. It uses the longest matching algorithm based on the Thai dictionary.

Next, from the words obtained, we transform them into classes and relations. This uses the text file in the storage level to help. Then the query is generated from classes and relations. The query is in the form of SPARQL. To handle the SPARQL query, JENA API and Apache Tomcat is used.

3. User level. The user inputs question from the web browser.

To begin with, the system will take the question in Thai from the user. The question is tokenized using Lexto which is based on the longest matching and general dictionary. Also we add the new set of vocabularies related to the cat features in our system. Then, we take the words to compare to our text files of class names and relations to categorize the keywords. If the keyword is in the category which is an Equivalent of relations (Relation I), we find the equivalentProperty of that relation for querying. After comparing, we take the matching keywords and their types. Two types are allowed: class and relation. We use class name, class type, relation name, relation type to query next.

For example, with the Thai query, "แมวขาวปลอดมีขนสีอะไร". Equivalently, "What is the hair color of the Khoa Pload cat?". The Thai sentence is tokenized into

" แมว | ขาวปลอด | มีขนสี | อะไร "

by using the modified Lexiton and our vocabulary. Then, these words are used to compared to the text files to categorize the words. From the example, we obtain breed, color, relation II, etc. as shown in Table 2.

Table 2. Example 1

ขาวปลอด	Class name
	class type = Breed
มีขนสี	Relation type = Relation II
	Relation = HasHairColorOf

In another example, with the Thai query, "แมวพันธุ์ใดบ้างมีขนสีเหมือนพันธุ์ขาวปลอด". This case contains the keyword "เหมือน" or "similar to" Then, again, these words are used to compared to the text files to categorize the words. From the example, we obtain breed, color, relation II, etc.

as shown in Table 3. Though the keywords are the same, we note here that there is a special word "similar to" in the query.

Table 3. Example 2

ขาวปลอด	Class name
	class type = Breed
มีขนสี	Relation type = Relation II
	Relation = hasHairColorOf

4. Processing Examples

In this section, we present several examples as well as the query mapping for each case. The first case is the case where the class type is the range. We may ask "แมวพันธุ์อะไรมีสีขาว". Equivalently in English, which cat has white color? The following query is used.

```
SELECT ? p
WHERE { ? p owl:equivalentClass _:x .
_:x owl:someValuesFrom test:classes.
_:x owl:onProperty test:relation}
```

In this case, the relation is hasHaircolorOf, the class is white. We find the node with the relation with some values are white. Then, apply the equivalent property to obtain the results.

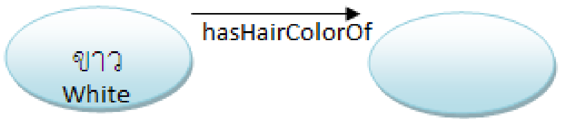


Figure 4. Example of using direct equivalent property

For another case, where the class type is the domain. There are two cases. The case where relation is type II.

```
SELECT ? p
WHERE { ? p owl:equivalentClass _:x .
 _:x owl:someValuesFrom test:classes
 _:x owl:onProperty _:c .
 test:relation owl:inverseOf _:c}
```

When the relation is type II, we use the query.

SELECT ? p

WHERE { ? p owl:equivalentClass _:x .

_:x owl:someValuesFrom test:classes.

_:x owl:onProperty _:c .

_:c owl:inverseOf test:relation }

With the above Thai query,"แมวขาวปลอดมีขนสีอะไร". It is type II. We obtain the keywords and types in Table 1. We take the value of hasHairColorOf to find the equivalent class. Then, the results are used with some values from the "ขาวปลอด" and find the equivalent result. The relation should be inversed for query as in Figure 5.

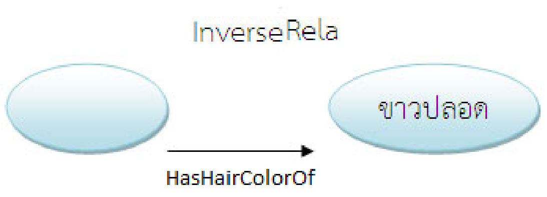


Figure 5. Example of using inverse property

The remainder cases are more complex where the class does not match with either domain or range. Consider the following query for the question "ถ้าแมวไข้สูงต้องรักษาอย่างไร". Equivalently, if the cat has a high fever, how to cure it?

SELECT ? p ? s

WHERE

? p owl:equivalentClass _:x .

_:x owl:onProperty ?o.

_:x owl:someValuesFrom test:classes

_:y owl:inverseOf test:relation

_:z owl:someValuesFrom ? p.

_:z owl:onProperty _:y .

? s owl:equivalentClass _:z .

? o rdfs:range test:typeClass

? o rdfs:domain _:c.

test:relation rdfs:domain _:c.

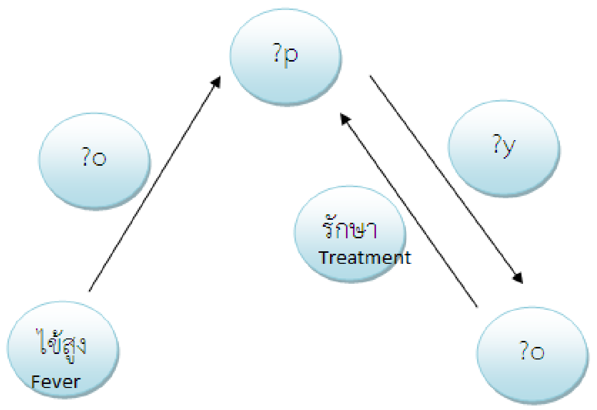


Figure 6. Example using two properties (I)

In this case in Figure 5, it is necessary to find the middle node (? p) between relation and class. This node is the domain of the relation type and any relation (? o). The relation has the range which is the class type in the keyword. Then, we take this relation to find the middle node. The middle node is used with the relation keywords to find the answer.

Example query is "ถ้าแมวไข้สูงต้องรักษาอย่างไร". Equivalently, if the cat has a high fever, how to cure it? Here, "fever" = Symptom and relation = "treatment". Both of them are not domain and range of Treatment. Thus, we take "treatment" to find the domain which is the same domain as the relation that connects to the symptom. Then, we take the relation to find the middle node. That is what is the middle node of "fever". Then, the result is taken to combine with "treatment" to find the answer.

The other case is where we find the class values in the keywords as in Figure 7.

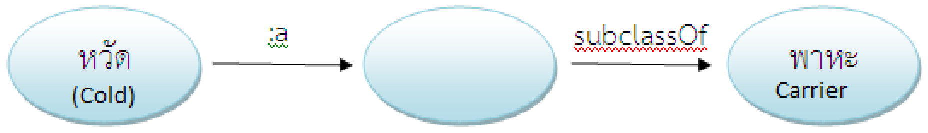


Figure 7. Example using two properties (II)

```
SELECT ? s
 WHERE { ? s rdfs:subClassOf test:typeClass.
 ? s owl:equivalentClass _:x .
 _:a rdfs:domain test:typeClass.
 _:x owl:onProperty _:a .
 _:x owl:someValuesFrom test:classes
```

The middle node needs to be discovered. It is the domain of the relation. Then, the relation is compared to class and that node is considered as the subclass of the class type.

For example, with the query, "อะไรเป็นพาหะของหวัด", equivalently, it is asked "what is the carrier of cold?". "cold" is the class. However, we don't find the relation. We find the category of the answer node which is of type "carrier". The answer node is the subclass of "carrier". The "carrier is the domain of any relation that is connected to "cold". Combining all, we obtain the answer node.

Similarly, when only relation name is found, we use the following query.

```
SELECT ? s
WHERE { ? s rdfs:subClassOf test:typeClass.
 ? s owl:equivalentClass _:x.
 _:x owl:onProperty test:relation}
```

The answer is the subclass of targeted class category as in Figure 8.

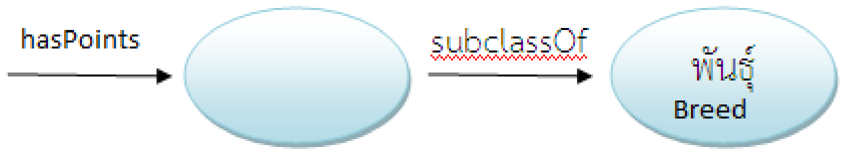


Figure 8. Example using subclass

The example query is "แมวพันธุ์อะไรมีแต้ม ". Equivalently, which cat breed has points? The relation is "has points". The class is not found but type of class is "Breed". Then, we find the subclass which is equivalent to the relation.

The last example is that there is the keyword "similar" in the sentence.

```
SELECT distinct ? p
WHERE { ? p owl:equivalentClass _:x .
 _:x owl:onProperty test:relation .
 _:x owl:someValuesFrom _:b .
_:c owl:inverseOf test: relation .
 _:b owl:equivalentClass _:a .
 _:a owl:onProperty _:c .
 _:a owl:someValuesFrom test:classes}
```

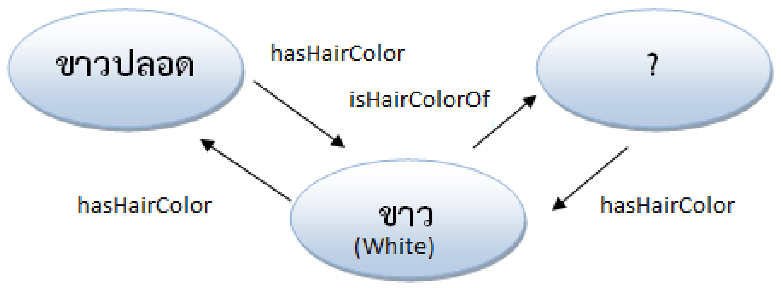


Figure 9. Example using two properties (III)

Example query is "แมวพันธุ์ใดบ้างมีขนสีเหมือนพันธุ์ขาวปลอด". Equivalently, which cat breed has the hair color similar to Khao Pload? We obtain relation = hasHairColor and classes = ขาวปลอด.

To query, we find the node in relationship with "hasHairColor". Then, we take the relation to check and find inverse of it where we get "isHairColorOf". Then, we take it to query for the answer of all colors of some values of Khao Pload. That is those are the hair colors of some Khao Pload. Then,we find the equivalent class X with the same color of Khao Pload. This is shown in Figure 9.

5. Results and Discussion

In this work, we only address the knowledge we represent. If more knowledge are added, the ontology structures changed, the cases of each question pattern may need to be reconsidered. However, if the knowledge addition does not affect, the ontology structure, the system can be used. We also provide the tool for updated keywords in the files as in Section 3.2.

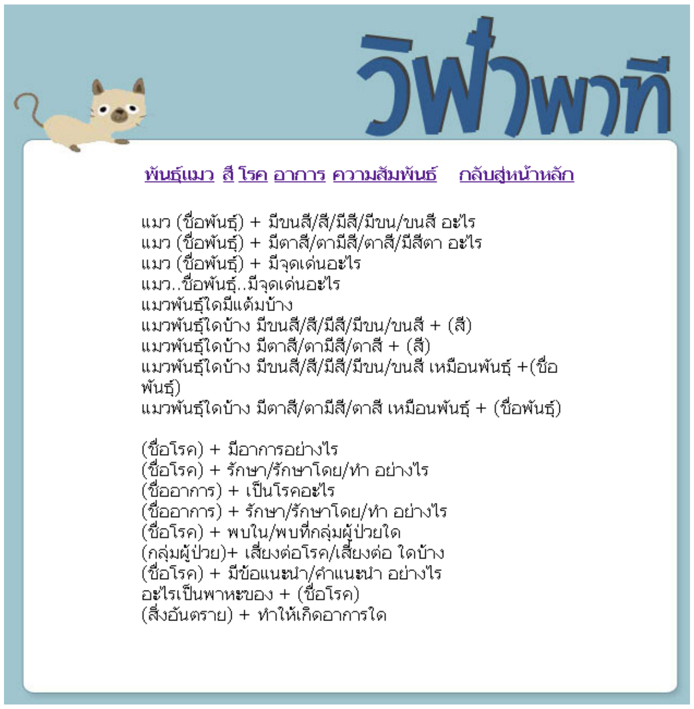
One of the problem due to Thai language is the keywords (verb) with synonyms. The system cannot check against the synonyms. For example, with the query asking "the cat has a fever" or "the cat fever". In Thai, "fever" can be a verb or a noun (where we use with "has"). The linguistic process is needed such as stemming words, finding synonym etc. to extend the efficiency of the system. Those issues are very subtle for Thai language.

Moreover, with this prototypes, many possible properties are not considered such as transitivity etc. A proper design with these properties will enhance the effectiveness.

Current test with 40 Thai sample questions reveals that the system can answer 32 questions and 31 questions are correct answer.

The prototype is developed using Java with the Jena middle ware. The ontology tool used (is Prototege 4.1. The application is in the form of web application with Apache Tomcat 6.1. Each component is Figure 3 (processing level) is implemented as Java module. Lexto is the tokenzing open source software where the API is provided by http://lexitron.nectec.or.th/2009_1/.

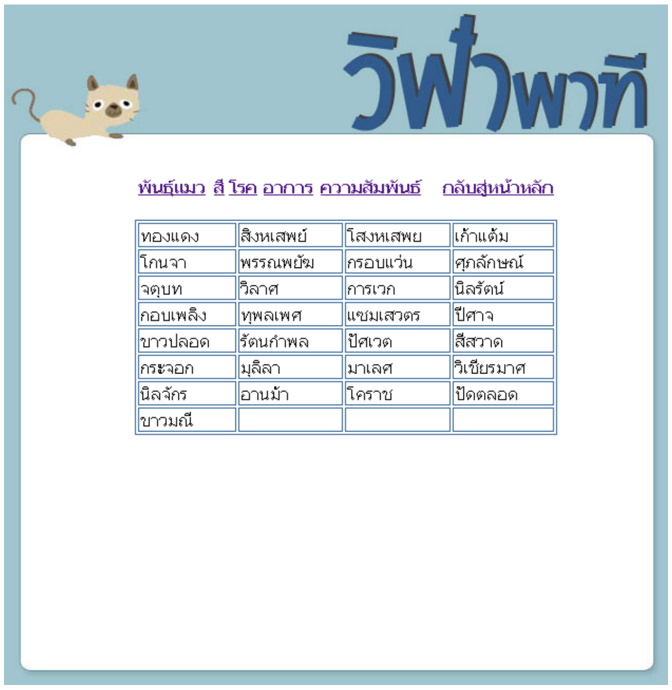
Figure 10 shows the example of the help menu of the system where we guide the example of question format. Figure 11 shows the directory of Thai cat breeds. Figure 12 shows the directory of the cat disease. The dictionary is stored as a text file where words can be added further.



แมว (ชื่อพันธุ์) + มีขนสี/สี/มีสี/มีขน/ขนสี อะไร
แมว (ชื่อพันธุ์) + มีตาสี/ตามีสี/ตาสี/มีสีตา อะไร
แมว (ชื่อพันธุ์) + มีจุดเด่นอะไร
แมว..ชื่อพันธุ์..มีจุดเด่นอะไร
แมวพันธุ์ใดมีแต้มบ้าง
แมวพันธุ์ใดบ้าง มีขนสี/สี/มีสี/มีขน/ขนสี + (สี)
แมวพันธุ์ใดบ้าง มีตาสี/ตามีสี/ตาสี + (สี)
แมวพันธุ์ใดบ้าง มีขนสี/สี/มีสี/มีขน/ขนสี เหมือนพันธุ์ +(ชื่อพันธุ์)
แมวพันธุ์ใดบ้าง มีตาสี/ตามีสี/ตาสี เหมือนพันธุ์ + (ชื่อพันธุ์)

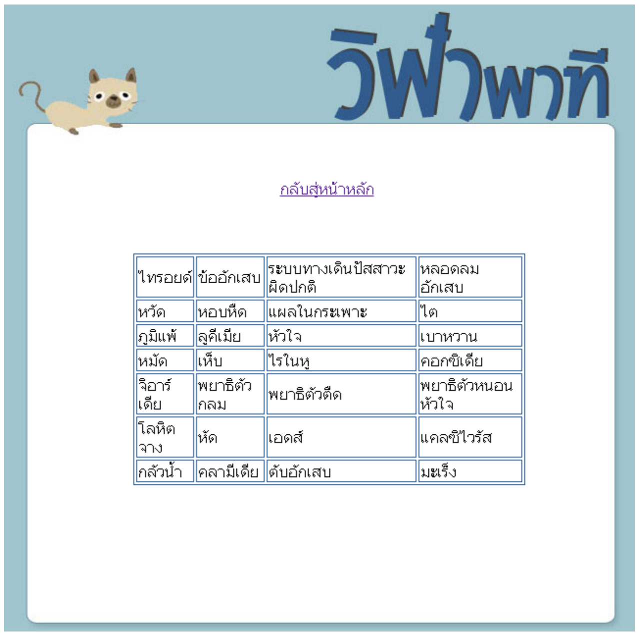
(ชื่อโรค) + มีอาการอย่างไร
(ชื่อโรค) + รักษา/รักษาโดย/ทำ อย่างไร
(ชื่ออาการ) + เป็นโรคอะไร
(ชื่ออาการ) + รักษา/รักษาโดย/ทำ อย่างไร
(ชื่อโรค) + พบใน/พบที่กลุ่มผู้ป่วยใด
(กลุ่มผู้ป่วย)+ เสี่ยงต่อโรค/เสี่ยงต่อ ใดบ้าง
(ชื่อโรค) + มีข้อแนะนำ/คำแนะนำ อย่างไร
อะไรเป็นพาหะของ + (ชื่อโรค)
(สิ่งอันตราย) + ทำให้เกิดอาการใด

Figure 10. The example question patterns



ทองแดง	สิงหเสพย์	โสงหเสพย	เก้าแต้ม
โกนจา	พรรณพยัฆ	กรอบแว่น	ศุภลักษณ์
จตุบท	วิลาศ	การเวก	นิลรัตน์
กอบเพลิง	ทุพลเพศ	แซมเสวตร	ปีศาจ
ขาวปลอด	รัตนกำพล	ปัศเวต	สีสวาด
กระจอก	มุลิลา	มาเลศ	วิเชียรมาศ
นิลจักร	อานม้า	โคราช	ปัดตลอด
ขาวมณี			

Figure 11. The breed of Thai cats



กลับสู่หน้าหลัก

ไทรอยด์	ข้ออักเสบ	ระบบทางเดินปัสสาวะผิดปกติ	หลอดลมอักเสบ
หวัด	หอบหืด	แผลในกระเพาะ	ไต
ภูมิแพ้	ลูคีเมีย	หัวใจ	เบาหวาน
หมัด	เห็บ	ไรในหู	คอกซิเดีย
จิอาร์เดีย	พยาธิตัวกลม	พยาธิตัวตืด	พยาธิตัวหนอนหัวใจ
โลหิตจาง	หัด	เอดส์	แคลซิไวรัส
กลัวน้ำ	คลามีเดีย	ตับอักเสบ	มะเร็ง

Figure 12. Dictionary of Thai cat disease

With the modification of the ontology, the new set of keyword files (Vocabulary category) can be rebuilt. The set of keywords for questioning (Dictionary) can be added to expand the question words for asking correspondingly. This storage level (in Figure 3) can be changed with minimum change at the processing level.

6. Conclusions

In this work, we demonstrate the development of question answering system using ontology as a knowledge representation. The case study is applied to Thai cat domain. We divide the work into to part which contains the front end and the back end part. The front end involves natural language processing in Thai. We use the open source tool such as WordNet and Lexiton to help tokenize. Also, specific keywords that are related to the domain are added. Then, the knowledge about Thai cats are gathers, cleaned and represented in an ontology form. The ontology design is also corresponding to the keywords extracted. The inference engine is based on Jena middleware. The whole architecture and design process is represented. The ontology can be further modified as well as the keywords can be updated to the system. The prototype system is developed using Java and Jena middleware as a web application.

In the future, the more patterns of questions can be considered. To generalize this, the regular expression and context free grammar can be applied for the formatting the questions. Also, more properties such as transitive, equivalent class, symmetric etc. can be added on when increasing more domain size.

References

[1] V. A. Lapshin, "Question-answering systems: Development and prospects", Automatic Documentation and Mathematical Linguistics, vol. 46, no. 3, **(2012)**, pp. 138-145.
[2] U. K. Sridevi and N. Nagaveni, "An ontology based model for document clustering", International Journal of Intelligent Information Technologies, vol. 7, no. 3, **(2011)**, pp. 54-69.

[3] M. A. Thomas, R. T. Redmond and V. Y. Yoon, "Using ontological reasoning for an adaptive e-commerce experience", International Journal of Intelligent Information Technologies, vol. 5, no. 4, **(2009)**, pp. 41-52.
[4] J. Kim and V. C. Storey, "Construction of domain ontologies: Sourcing the World Wide Web", International Journal Intelligent Information Technologies, vol. 7, no. 2, **(2011)**, pp. 1-24.
[5] B. Katz, S. Felshin, D. Yuret, A. Ibrahim, J. Lin, G. Marton, A. J. McFarland and B. Temelkuran, "Omnibase: Uniform access to heterogeneous data for question answering", 7th International Workshop on Applications of Natural Language to Information Systems; **(2002)** July 27-28; Stockholm, Sweden.
[6] M. E. Sucunuta and G. E. Riofrio, "Architecture of a question-answering system for a specific repository of documents", International Conference on Software Technology and Engineering; **(2010)** October 3-5; Puerto Rico, USA: IEEE, pp. 12-16.
[7] G. Attardi, A. Cisternino, F. Formica, M. Simi and A. Tommasi, "PiQASso: Pisa question answering system", In: Text Retrieval Conference (TREC); **(2001)** November 13-16; Maryland, USA: NIST, pp. 633-641.
[8] W. Jitkrittum, C. Haruechaiyasak and T. Theeramunkong, "QAST: Question answering system for Thai Wikipedia", Workshop on Knowledge and Reasoning for Answering Questions; **(2009)** August 2-9; Singapore, pp. 11–14.
[9] A. Lampert, "A quick introduction to question answering", **(2004)**.
[10] W. Lehnert, "A Conceptual theory of question answering", In Grosz BJ, Sparck Jones K, Webber BL, eds. Natural Language Processing, Los Altos, CA, USA: Kaufmann, **(1986)**, pp. 651–657.
[11] D. Brunner and S. Stall, "The Cat Owner's Manual: Operating Instructions, Troubleshooting Tips, and Advice on Lifetime Maintenance", Philadephia, USA: Quirk Book, **(2004)**.
[12] G. Bailey, "What Is My Cat Thinking?: The Essential Guide to Understanding Pet Behavior", San Diego, USA: Thunder Bay Press, **(2002)**.
[13] A. W. Kate, V. Roby and L. Southam, "The Pill Book Guide to Medication for Your Dog and Cat", Richmond, Texas, USA: Bantam, **(1998)**.
[14] N. Manippan, "Thai Cat", Bangkok, Thailand: Prasanmitra, **(1999)**.
[15] S. Horridge, S. Jupp, G. Moulton, A. Rector, R. Stevens and C. Wroe, "A Practical Guide To Building OWL Ontologies Using Prot´eg´e 4 and CO-ODE Tools", **(2007)**.