A Conceptual Model of Indonesian Question Answering System based on Semantic Web

A Conceptual Model of Indonesian Question Answering System based on Semantic Web

M Syarief; W Agustiono; A Muntasa; M Yusuf

University of Trunojoyo Madura, Jl. Raya Telang PO BOX 2 Kamal Bangkalan

Mohammad.syarief@trunojoyo.ac.id

Abstract. This study aims to develop a conceptual model of question answering system in Bahasa using Semantic web technology. A trigger was built to periodically collect data scattered on the internet in a data center to shorten the query process on data without reducing the data completeness. The test case domain was automotive. This study used four types of ontologies: main ontology, ontologies from manufacturers, dealers, and rentals. This research has some novelties, such as solving the weakness of distributed and centralized data and combining the advantages of both by storing distributed data into a pool (centralized data) periodically, then querying the pool. Furthermore, it makes contribution by providing a conceptual model of Indonesian question answering system, particularly in automotive domain. This has some implications, such as extend the body of knowledge of computer science, especially semantic web and ontology fields.

1. Introduction

Information retrieval system that is most often found is search engines, such as Google, Yahoo, and Bing. Searching with a search engine will display a list of possible answers link. A user still have to explore each link in the list to get what they want.

Unlike search engines, the question-answering system (QAS) will provide answers to questions directly, without tracing the list of article snippets as search engine results [1]. Therefore, in order to provide the right data in accordance with the information that user wants to find, the question-answering system is more appropriate to use than the search engine.

Besides data acquisition techniques, data (information) is also the most important part of the information search process. Sometimes user uses common words in keywords, where the word is not even in the available data set, so the search results do not meet the the user wishes, as in the question "what is the cheapest 2013 vehicle?". Usually, word ‘vehicle’ is not mentioned in outlet database. Generally, records in the outlet database refer to vehicle detailed specifications, such as Supra X, Freed, Jazz, Yaris, and so on. If the system fails to detect that Supra X is a part of motor, and motor is part of vehicle, it is very possible that the system answer is not in line with the user's expectations. Semantic web technology is needed to overcome this. Semantic Web [2] provides the technology and standards needed to add meaning that can be understood by the machine (machine-understandable) on the web in general, so that the computer can make an inference process against its data [3].

Based on the origin of the data, in general, data can be classified into two groups: centralized data and distributed data. The advantage of centralized data is that it speeds up the data retrieval process. In addition, in private data, the original data will only be owned by the owner or data storage provider.

However, in public data, especially for promotional purposes, the data owner wants the data to be as widely spread as possible. In public data, centralized data is not the best choice.

Public data is more appropriate using a distributed data model, so that other parties, such as search engines and QA systems can use the data according to its original format. However, in contrast to centralized data, data retrieval on distributed data takes longer because the engine performs remote queries.

This study will discuss the conceptual model and architecture of the Indonesian Question Answering System based on the Semantic Web. In order to overcome the shortage of centralized data and distributed data, this study will combine the advantages of both by storing distributed data into a pool (data center) periodically, then querying the pool.

One of distributed data implementations` is triple data in semantic web, such as RDF, OWL, Turtle, etc. However, triple data requires ontology as the data framework. Ontology can reusable. That is, all parties can use, add, or even make adjustments to the existing ontology to make it more suitable for use in our data.

The test cases domain that will be discussed in this paper is automotive. To facilitate mapping without translation, Indonesian ontology takes precedence. In this case, we choose to use OntoKendaraan [4] as the main ontology. This system is called AQAS (Automotive Question Answering System).

There have been many previous studies that discuss the QAS architectures. There were architecture of QAS built for the predicting the best answer [5], QAS with dialog models [6], QAS focused on image recognition [7]–[9], QAS with deep learning [10]–[13], medical QAS [14]–[16] , and YodaQA [17], [18]. Each QAS mentioned above has an unique architecture. so it's difficult to compare with one another. But the QAS that addresses the approach to providing exact answers [19] has the most similar features with this study. The differences are the methods used and the data source. The data source of this study came from various sources scattered on the internet, while [19] used unstructured data, such as paragraphs in English.

2. Research Methods

As an initial step in developing AQAS, this study needs to formulate a model that facilitates the features available in AQAS. This model will be a reference in analyzing and designing AQAS.

There are 2 main functions of AQAS: collecting data knowledge in data center and obtaining answers based on user question. The data knowledge uses OWL/XML or RDF/Turtle form. Usually the data comes from manufacturers, dealers, and rentals scattered on the internet.

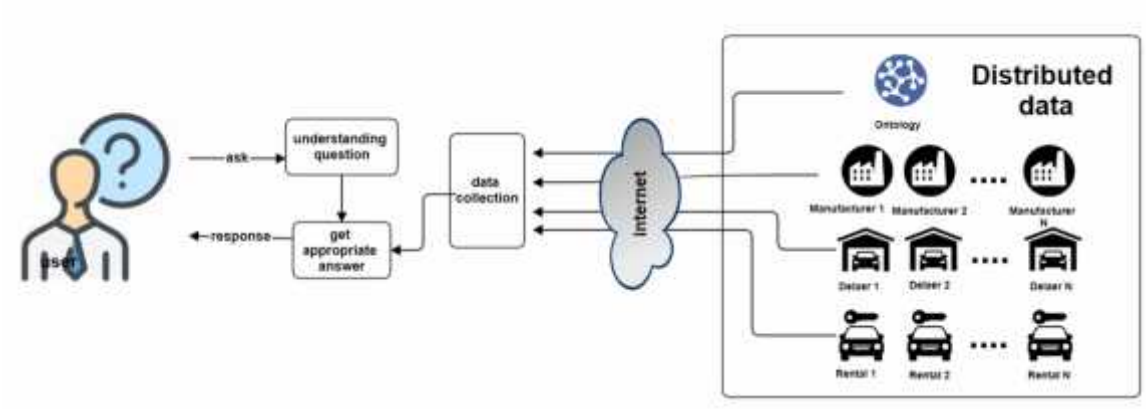

Fig 1. Conceptual model of AQAS

Fig 1 is a conceptual model that can illustrate both AQAS functions in a simple way. There are 3 main elements in the model:

1. Users, who can ask something to AQAS using Indonesian. The user is entitled to get a response to the question he asked.
2. The system that has a task: understanding user questions, looking for information related to these questions, and providing responses to these questions.
3. Distributed data, which is data that is spread on the internet and will be a reference in answering user questions.

The answering workflow is illustrated in Fig. 2.

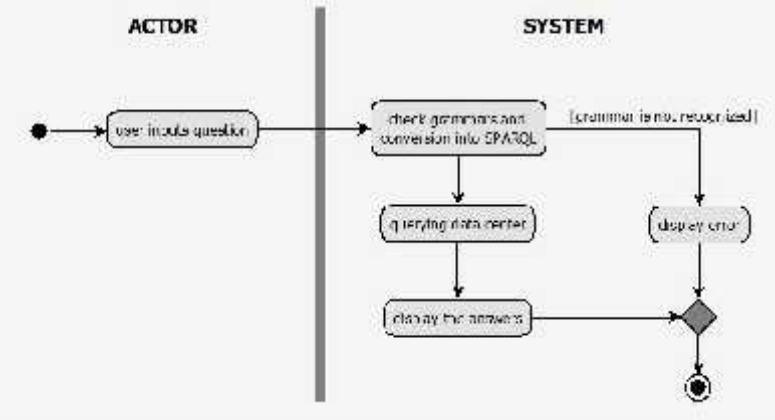

Fig 2. The AQAS answering workflow

3. Results and Discussions

In this section, we will discuss more details about the AQAS design based on Fig. 1. Moreover, AQAS needs five components based on the main functions to operate properly, such as system interface, NLP server, data server (data center), data loader, and scattered data knowledge. The relationship of the components is illustrated in the AQAS architecture as shown in Fig. 3.

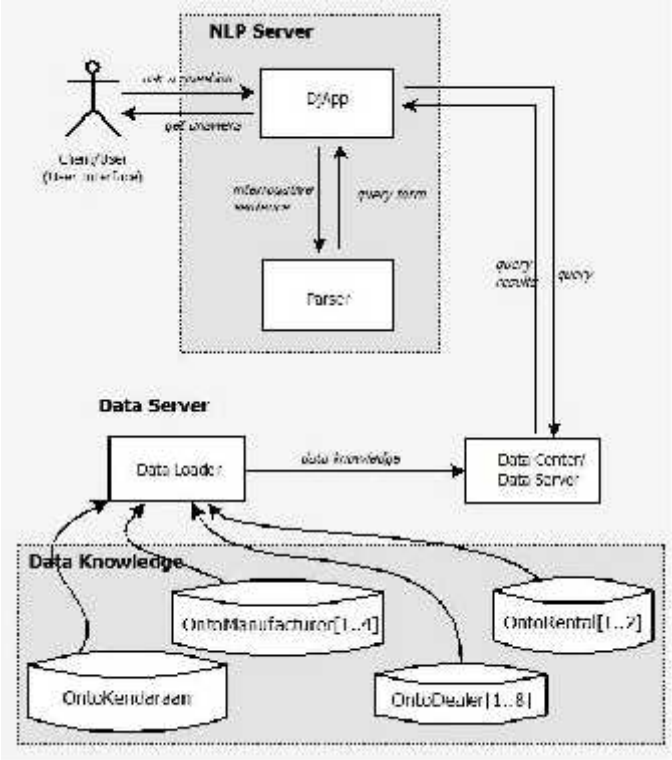

Fig. 3. AQAS architecture

3.1. System Interface

The system interface is a system display that directly communicates with users, that is a web form display where user enters the question and the system response display to the question.

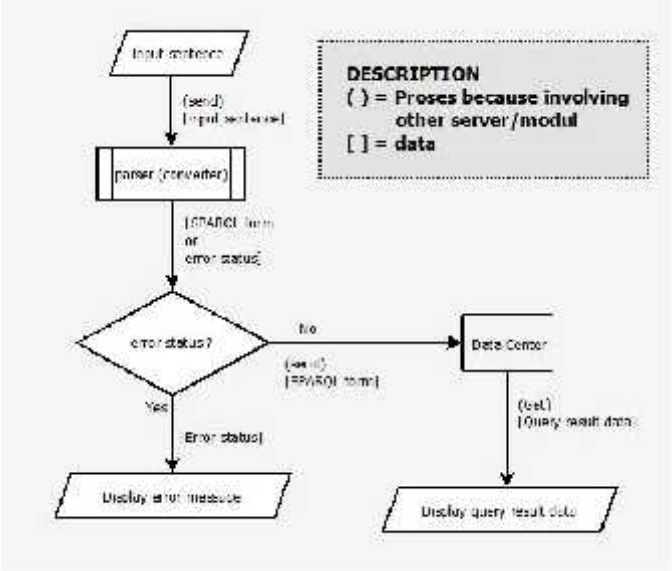

Fig. 4. Processes handled by DjApp

If the question is recognized, then the system response will be a phrase or list of phrases, according to the acquisition of query results in the semantic data center. However, if it is wrong, then the system response will be failure status and/or equipped with information that causes failure or advice to be recognized sentence format).

3.2. NLP Server

In Fig. 3, there is *DjApp* (Django Application) and parser in the NLP server block. *DjApp* is a web application that is built using Django framework (Python Programming Language), while the parser is a module used by *DjApp* to convert input sentence into SPARQL form. *DjApp* handles interfaces with users, input processing, query delivery, and query results, as shown in Fig. 4.

DjApp communicates with data center using web service technology to get data needed by users. DjApp sends SPARQL queriey to data center using an external library: SPARQLWrapper.

Parser is a special module used by *DjApp* to check sentence grammar and transform input sentence into SPARQL form. Checking grammatical sentence are intended to understand question more easily.

Natural language processing (NLP) occurs in NLP server (sub unit: Parser) and has the purpose to check the grammar of input sentences, and then describe the question sentence into the formulation of SPARQL queries. To obtain these objectives, a top-down parsing approach based on sentence predicate recognition is used. Complex sentence recognition is performed by breaking complex sentence into several simple sentences, then conversion into SPARQL form is performed on each of these simple sentences.

3.3. Data server/ data center

The main function is the data center as a storage medium of knowledge in triple forms (triple store). In addition to the triple store, the data center should also have a reasoner to make inferences about the data set.

Inference capabilities required in AQAS:

1. Inheritance of subClass status

Example of inheritance of sub-class status: if A is a sub-class of B, and B is a sub-class of C, then A must be a sub-class of C (Fig. 5).

```
premise 1 : ahm:revo rdf:type kendaraan:Motor
premise 2 : kendaraan:Motor rdf:type kendaraan:Kendaraan
-----------------------------------------------------------
conclusion: ahm:revo rdf:type kendaraan:Kendaraan
```

Fig 5. subClass inheritance

2. Inheritance of subProperty status.
   Example of subProperty status inheritance: if property A is a subProperty of property B, and C is an instance that has property A with value D, then C must also have property B with value D (Fig. 6).

```
premise 1: kendaraan:remDepan rdfs:subPropertyOf kendaraan:rem
premise 2: ahm:revo kendaraan:remDepan ahm:Cakram
-------------------------------------------------------------
conclusion: ahm:revo kendaraan:rem ahm:Cakram
```

Fig 6. subProperty inheritance

3. Instance equivalence.
   If two instances are equivalent, they will both have the same property and the value of the property can be exchanged (Fig. 7).

```
premise 1 : ahm:backbone kendaraan:nama "backbone/tulang punggung"
premise 2 : yamaha:underbone kendaraan:nama "underbone"
premise 3 : ahm:backbone owl:sameAs yamaha:underbone
-----------------------------------------------------------------
conclusion 1 : yamaha:underbone kendaraan:nama "backbone/tulang punggung"
conclusion 2 : ahm:backbone kendaraan:nama "underbone"
```

Fig 7. Instance equivalence

4. Property equivalence.
   If two properties are equivalent, they can be exchanged (Fig. 8).

```
diketahui : (a) kendaraan:kapasitasMesin rdf:type owl:ObjectProperty
            (b) kendaraan:volumeLangkah rdf:type owl:ObjectProperty
Premise :     (a) kendaraan:kapasitasMesin owl:equivalentProperty
                  kendaraan:volumeLangkah
              (b) ahm:revo kendaraan:kapasitasMesin 110
-------------------------------------------------------------------
conclusion: ahm:revo kendaraan:volumeLangkah 110
```

Fig 8. Property equivalence

Data centers will be built using the OpenRDF Sesame (Java) framework. Even though OpenRDF Sesame has its built-in reasoner, it still does not meet the inference capabilities required by the system, as mentioned above. Therefore, the reasoner used in AQAS is OWLIM. OWLIM is implemented in Java and is packaged as a Storage and Inference Layer (SAIL) for the Sesame OpenRDF framework.

3.4. Data loader

Data loader is a standalone application that aims to collect knowledge data into one in the data center. The data comes from several files that are spread on the internet with locations that have been previously known, both in the form of ontology (schema), and a collection of instances. The library used in the data loader is httplib2. Httplib2 is used to keep HTTP 1.1 active and keep the socket open, so it possible to make multiple requests over the same connection.

As the knowledge data to be collected is not transaction data, no real time data update is needed. Therefore, data updates at the data center will be conducted periodically. This can be implemented using the default operating system scheduling program (Task Scheduler on Windows or crontab on Linux).

3.5. Knowledge Data

This knowledge data can be divided into 4 groups:

1. OntoKendaraan [4].

OntoKendaraan is the main ontology used as a reference ontology in the development of other ontologies. OntoKendaraan based on OWL DL standard.

2. OntoManufacturers.
   OntoManufacturer is an ontology and a collection of motorized vehicle instances provided by the manufacturer.
3. OntoDealers.
   OntoDealer is a file that contains a collection of instances provided by motor vehicle sales service providers (dealers).
4. OntoRentals.
   OntoRental is a file that contains a collection of instances provided by motor vehicle rental.

4. Conclusion and further work

This study has identified that AQAS needs five components based on the main functions to operate properly, such as system interface, NLP server, data server (data center), data loader, and scattered data knowledge. Furthermore, knowledge data can be divided into 4 groups, i.e. OntoKendaraan, OntoManufacturers, OntoDealers, and OntoRentals.

This has some implications, such as extend the body of knowledge of computer science, especially semantic web and ontology fields. The limitation of this research is that the data should not be transaction data which must stay updated. The future research will add optimization of the rule-based feature in triple store inference technique and implementation of data using Notation-3 format by optimizing rule-based feature to lighten up the parser work.

Acknowledgments

The authors would like to thank Mr. Khabib Mustafa as a thesis supervisor at the Department of Computer Science, UGM.

References

[1] C. Kwok, O. Etzioni, and D. S. Weld, “Scaling question answering to the web,” *ACM Trans. Inf. Syst.*, vol. 19, no. 3, pp. 242–262, 2001.
[2] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web,” *Sci. Am.*, vol. 284, no. 5, pp. 34–43, 2001.
[3] L. Yu, *A Developer’s Guide to The Semantic Web*. Springer, 2011.
[4] M. Syarief, “Ontomotif: Ontologi Pencarian Informasi Kendaraan Bermotor,” in *Seminar Nasional Sains dan Teknologi*, 2015.
[5] D. Elalfy, W. Gad, and R. Ismail, “A hybrid model to predict best answers in question answering communities,” *Egypt. Informatics J.*, vol. 19, no. 1, pp. 21–31, 2018.
[6] C. Chakrabarti and G. F. Luger, “Artificial Conversations for Customer Service Chatter Bots : Architecture , Algorithms , and Evaluation Metrics,” pp. 1–41.
[7] C. Xiong, S. Merity, and R. Socher, “Dynamic Memory Networks for Visual and Textual Question Answering,” vol. 48, no. 2015, 2016.
[8] H. Xu and K. Saenko, “Ask , Attend and Answer : Exploring.”
[9] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, “Stacked Attention Networks for Image Question Answering,” no. 1, pp. 21–29.
[10] B. Z. Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, “Applying Deep Learning to Answer Selection: A Study and an Open Task,” *IEEE Work. Autom. Speech Recognit. Underst.*, pp. 813–820, 2015.
[11] Y. Sharma and S. Gupta, “Deep Learning Approaches for Question Answering System,” *Procedia Comput. Sci.*, vol. 132, pp. 785–794, 2018.
[12] B. X. & B. Z. Ming Tan, Cicero dos Santos, “LSTM- BASED DEEP LEARNING MODELS

FOR NON- FACTOID ANSWER SELECTION," in *International Conference on Learning Representations*, 2016, no. 1, pp. 1–11.
[13] M. Tan, C. Santos, B. Xiang, and B. Zhou, "Improved Representation Learning for Question Answer Matching," pp. 464–473, 2016.
[14] M. Sarrouti, S. Ouatik, and E. Alaoui, "A passage retrieval method based on probabilistic information retrieval model and UMLS concepts in biomedical question answering," *J. Biomed. Inform.*, vol. 68, pp. 96–103, 2017.
[15] Y. Cao *et al.*, "AskHERMES : An online question answering system for complex clinical questions," *J. Biomed. Inform.*, vol. 44, no. 2, pp. 277–288, 2011.
[16] T. R. Goodwin and S. M. Harabagiu, "HHS Public Access," 2017.
[17] P. Baudiˇ and J. Sediv, "Modeling of the Question Answering Task in the YodaQA System," pp. 4–9.
[18] P. Baudiˇ, "YodaQA : A Modular Question Answering System Pipeline," pp. 1–8, 2015.
[19] R. Barskar and G. F. Ahmed, "Procedia Engineering An Approach for Extracting Exact Answers to Question Answering (QA) System for English Sentences," vol. 30, pp. 1187–1194, 2012.