



VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Siheng Liu

Question answering system of a specific
domain and its implementation on a NAO
humanoid robot

Technology and Communication
2018

FOREWORD

This is my final thesis for the Degree Program in Information Technology in Vaasan Ammattikorkeakoulu (Vaasa University of Applied Science). I have studied in this school for almost two years and during this period, I have come to know many friends and teachers who have helped me a lot. It was a chance of a lifetime for me to have a such memorable experience in Finland.

First of all, I would like to thank my supervisor, Dr. Yang Liu. He spent a lot of time working on my thesis and provided many useful suggestions. Without him, I may not had the opportunity to study here and got a better platform to make progress.

I would take the opportunity to thank the whole IT department of VAMK, especially Dr. Chao Gao, Dr. Ghodrat Moghadampour, Mr. Santiago Chavez Vega, Dr. Seppo Mäkinen, Dr. Smail Menani. You guided me and encouraged me during these two years of studying at VAMK and provided us with high quality courses.

Also, I want to express my deep gratitude to my parents who always support my decision financially and emotionally. I love them. And I am thankful all my friends at the Botnia RoboCup laboratory, I wish you all good luck.

Siheng Liu

Vaasa, Finland

24.03.2018

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Program in Information Technology

ABSTRACT

Author	Siheng Liu
Title	Question answering system of a specific domain and its implementation on a NAO humanoid robot
Year	2018
Language	English
Pages	52
Name of Supervisor	Yang Liu

This thesis introduces the development of a question-answer system in a specific domain and its implementation on the Nao robot. The goal of this paper is to build a limited domain, high-performance and speech supported automatic question-answering system based on NAO, the humanoid robot for hospital information.

This thesis is divided into four parts, web information extraction, the construction of the knowledge base, natural language processing and speech recognition. The data used for building a knowledge base is gathered from the official website of the hospitals by using web crawler. Then the data is stored in a database and converted to RDF type. In the second part, the questions which are asked by the users need to be converted into SPARQL to search the answer from the knowledge base. The last part is connected with the Nao robot, a recorder on Nao is used to record the voice of the users and the Google Cloud Speech API is used in this thesis for speech recognition to transfer the voice file into text. After getting the answer, the speech is synthesized by the Nao robot to interact with the user.

This thesis has achieved to use the Nao robot to answer users' question about the some aspects of hospitals in Finland. The location, telephone number, open time can be queried.

Python is the programming language used to develop this project. The code testing is done under macOS 10.13.1, Python version 2.7, and Nao robot vision 5.

Keywords Question Answering System, NAO robot, Question analysis

CONTENTS

ABSTRACT

LIST OF ABBREVIATIONS	6
LIST OF FIGURES AND TABLES	7
1 INTRODUCTION	9
1.1 Background.....	9
1.2 Introduction to question answer system.....	9
1.3 Purpose.....	11
1.4 Overview Structure	12
2 TOOLS AND TECHNOLOGIES	13
2.1 Programming Tools.....	13
2.1.1 Python	13
2.1.2 MySQL	13
2.2 Web crawler	13
2.3 RDF & OWL.....	14
2.4 Tools used to build a Knowledge Graph.....	15
2.4.1 Protégé	15
2.4.2 D2RQ	15
2.4.3 Apache Jena	15
2.5 Natural Language Toolkit	16
2.6 Speech Recognition	17
2.7 NAO Robot.....	18
2.7.1 Introduction to Nao Robot	18
2.7.2 NAOqi.....	20
3 OVERALL DESIGN	22
3.1 Structure of the whole project.....	22
3.2 Flowchart of the whole project	22
4 WEB INFORMATION EXTRACTION	24
4.1 Web Crawler	24
4.1.1 Preparation	24

	5
4.1.2 Libraries	24
4.2 Information Extraction.....	25
4.3 Insert into Database.....	30
5 KNOWLEDGE BASE CONSTRUCTION.....	32
5.1 Ontology modeling	32
5.2 Relational database to RDF	35
5.3 Apache Jena SPARQL endpoint and reasoning.....	37
6 QUESTION TO SPARQL	40
6.1 Natural Language Parsing.....	40
6.2 Query Generation & DSL.....	41
7 IMPLEMENTATION ON NAO ROBOT	44
7.1 Speech Recognition	44
7.2 Result	47
8 RECOMMONDATION FOR FUTURE RESEARCH.....	49
9 CONCLUSION.....	50
REFERENCES	51

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
QA	Question Answering
SPARQL	SPARQL Protocol and RDF Query Language
KB	Knowledge Base
RDF	Resource Description Framework
OWL	Web Ontology Language
RDFS	RDF Schema
API	Application Programming Interface
NLTK	Natural Language Toolkit
POS	Part of Speech
ASR	Automatic Speech Recognition
STT	Speech to Text
TTS	Text to Speech
HTML	Hypertext Markup Language
DOM	Document Object Model
GMO	Genetically Modified Organism
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
W3C	The World Wide Web Consortium
REFO	Regular Expressions for Objects
DSL	Domain Specific Language
FTP	File Transfer Protocol

LIST OF FIGURES AND TABLES

Figure 1. QA System	p.11
Figure 2. Web Crawler/4/	p.14
Figure 3. Structure of RDF	p.14
Figure 4. Apache Jena/5/	p.16
Figure 5. Parse tree generated with NLTK	p.17
Figure 6. Speech Recognition Comparison /8/	p.18
Figure 7. Nao Robot /14/	p.19
Figure 8. Specifications of NAO/15/	p.20
Figure 9. The NAOqi process/17/	p.21
Figure 10. Whole structure	p.22
Figure 11. Flowchart	p.23
Figure 12. HTML Example of Website	p.26
Figure 13. HTML Element List	p.27
Figure 14. HTML	p.28
Figure 15. HTML DOM Tree	p.28
Figure 16. Information on the Website	p.29
Figure 17. Sample HTML	p.30
Figure 18. Database	p.31
Figure 19. ER Figure of Database	p.32
Figure 20. IRI	p.33
Figure 21. Classes	p.33
Figure 22. Object Property	p.34
Figure 23. Data Properties	p.34

Figure 25. NT file	p.36
Figure 26. RDF to TDB	p.37
Figure 27. Fuseki Sever	p.38
Figure 28. Browser access to Fuseki Sever	p.39
Figure 29. Connect Fuseki Sever	p.39
Figure 30. Semantic matching	p.41
Figure 31. Domain Specific Language	p.42
Figure 32. Interpret Method	p.42
Figure 33. Control Process of NAO Robot/16/	p.44
Figure 34. Audio Recorder	p.45
Figure 35. Transfer wav File	p.45
Figure 36. Google Cloud Speech API	p.46
Figure 37. Google Cloud Speech API Code	p.47
Figure 38. Nao Robot	p.48
Figure 39. Result	p.48

1 INTRODUCTION

1.1 Background

Since entering the 21st century, the Internet has entered every aspect of people's lives. By connecting to the Internet, everyone can publish their own viewpoints, inquire about desired information, and learn to use knowledge. With more and more people using the internet, the amount of information is growing rapidly, and we really enter the era of information explosion. It becomes easy for people to search almost everything through major search engines such as Google, Bing, and Baidu.

However, the current search engine is mainly based on keyword matching, for example, the question “How long is the great wall of China?”, a traditional search engine searches a web page with corresponding information based on a few keywords to a large number of web pages and then presents the user with a list of related links. The users have to read and analyze themselves which information is useful for them. And even if the information found on the website is related, the correctness of this information is still difficult to guarantee. The Question Answering System solves this problem. It provides the only and correct answer based on its knowledge base.

The current intelligent question answering system is mainly divided into two aspects: Open Domain QA, and Limited Domain QA. Open Domain QA requires a large-scale knowledge base and it needs a large amount of human and financial resources to establish a comprehensive knowledge base. In contrast to the open system of question and answer, the use of limited areas of question and answer system is more extensive and mature. Among them, there are representative systems such as expert systems, which limit the scope of questions, and are usually restricted to specific areas such as medical treatment, physical education, culture, education, etc./1/

1.2 Introduction to question answer system

The question answering system is a computer science discipline in the field of information retrieval and natural language processing (NLP). It focuses on establishing a system that automatically answers questions raised by humans in natural language.

A QA implementation, usually a computer program, can construct answers by querying a structured knowledge or information database, usually a knowledge base. More commonly, QA systems can extract answers from unstructured sets of natural language documents.

QA research attempts to deal with a variety of problem types, including facts, lists, definitions, methods, causes, assumptions, semantic limitations, and cross-linguistic issues.

Closed domain question answering is a much easier task to deal with in specific domains, such as medicine or vehicle maintenance, because NLP system can use the domain-specific knowledge that is often formalized in an ontology. Or, the closed domain may mean that only a limited type of problem is accepted, such as the question of asking for descriptive rather than procedural information/2/.

Intelligent question answering system is a very popular research direction in the field of natural language processing and machine learning. The use of implementation processes and technologies varies according to the sources of data and knowledge in different areas. In general, there are many differences. It can be divided into question analysis, information extraction and answer sorting. Question analysis is mainly to analyze the user input query, classify it correctly, know the user's intention, extract the keywords of a query, and expand the keywords.

Information extraction is after the above question analysis completed, then the corresponding knowledge base, document retrieval, find the corresponding contents paragraphs, if the question answering system is based on the Web, it is through the traditional search algorithm using the specific model of the calculation of each candidate answer these as a basis for the next step of the engine to retrieve the answer ranking to document relevant information. Figure 1 shows the normal processes of a QA system.

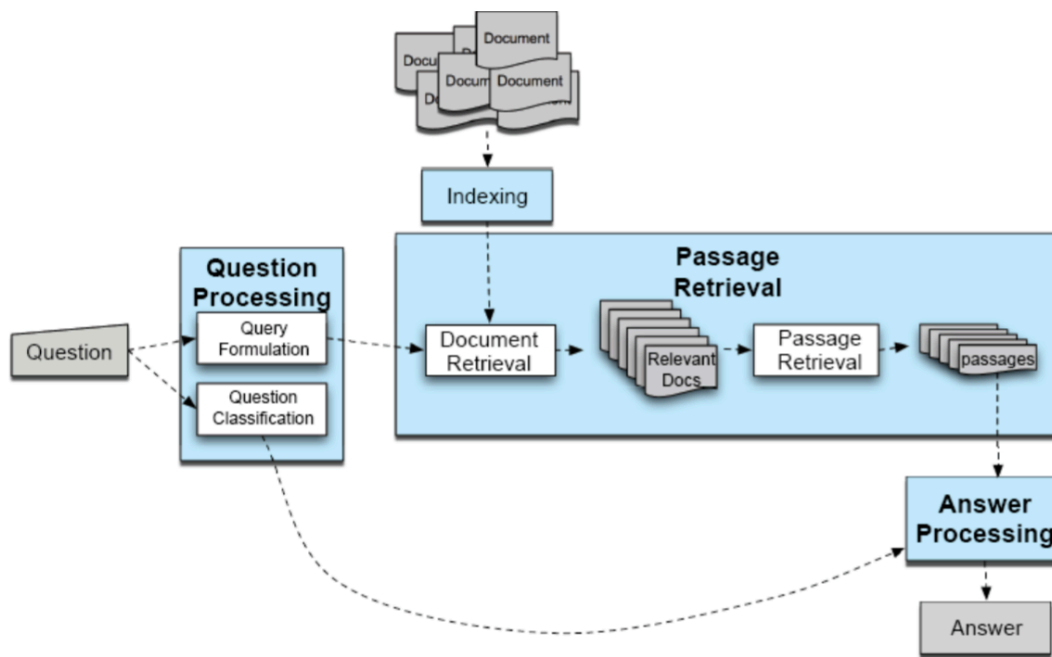


Figure 1. QA System

1.3 Purpose

Based on the research background, current situation and challenging questions of the above QA system, the goal of this paper is to build a limited domain, high-performance and speech support automatic question answering system based on the NAO humanoid robot for hospital information in Finland. In order to achieve this purpose, a list of works is done to complete the system. First of all, the automatic crawling technology is used to get hospitals' information from their websites such as unit name, location, telephone number and open time. Secondly, we extract knowledge and extract data from page information, and complete a relatively complete knowledge base as the basis for future research. Then, the automatic question answering system based on the knowledge base, including the analysis of questions and the retrieval of the answers. In this step, the natural language is converted to SPARQL to query. Finally, based on the automatic question and answer of the text, further utilized speech recognition, and speech synthesis techniques are added on Nao robot. This paper proposes to use the cloud speech recognition method to overcome the limitation of Nao's own insertion recognition engine.

1.4 Overview Structure

This thesis mainly consists of nine chapters. The First chapter is an introduction to this paper and it introduces relevant background knowledge, the purpose and structure of this thesis. In chapter two, tools and technologies which are used to complete this project are introduced briefly, including the programming language, web crawler, NLTK, speech recognition and the Nao robot. Chapter three gives an overall design of the whole project, it shows the structure and sequence of the whole project which is quite clear. The fourth chapter is mainly about web crawler, it introduces how can we use beautiful soup and a regular expression to extract information from a webpage and store them in a database. Knowledge about how to construct a knowledge base is written in chapter five, several tools are used to modeling and configure the knowledge base. Chapter six introduces how to convert a natural language question into SPARQL to search an answer from the knowledge base. The seventh chapter describes the connection with the Nao robot and how to use the speech recognition API. Chapter eight briefly describes the limitation of this project and give some recommendations and expectancies for future research. The last chapter is the summary and conclusion of the thesis.

2 TOOLS AND TECHNOLOGIES

2.1 Programming Tools

2.1.1 Python

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. In this thesis python version 2.7 is used to program/3/. PyCharm is the Python IDE used in this thesis to manage a python project.

2.1.2 MySQL

MySQL is one of the most popular relational database management systems, in the WEB application MySQL is one of the best Relational Database Management System (RDBMS) application software. The RDBMS can be used to store and manage huge volume of data. Those data can be stored into different tables and these tables can be related by primary keys or other keys known as foreign keys. Also, MySQL can be allowed on multiple systems and supports multiple languages. These programming languages include C, C ++, Python, Java, Perl, PHP, Eiffel, Ruby and Tcl.

2.2 Web crawler

A web crawler starts with a list of URLs to visit, called the seeds. Figure 2 shows how it works. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies. If the crawler is performing archiving of websites it copies and saves the information as it goes. The archives are usually stored in such a way they can be viewed, read and navigated as they were on the live web, but are preserved as 'snapshots'. /4/

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work. Compared with other tools, beautiful soup is more convenient and time-saving.

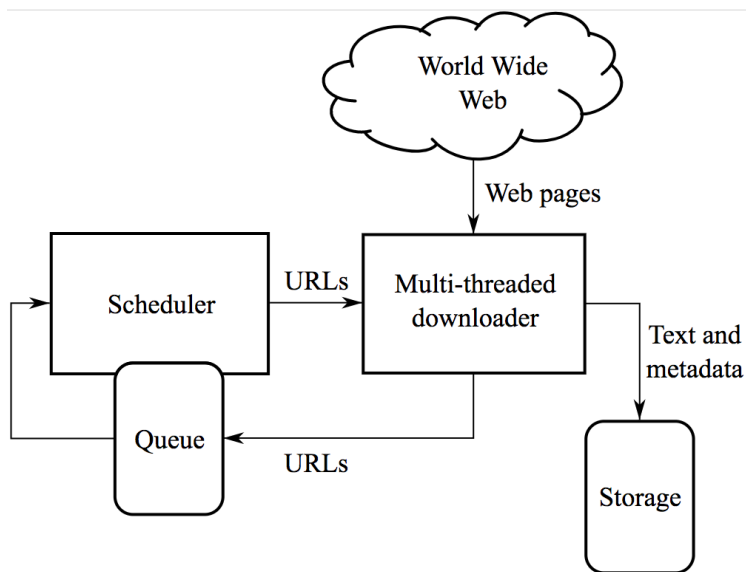


Figure 2. Web Crawler/4/

2.3 RDF & OWL

RDF (Resource Description Framework) is essentially a data model. It provides a unified standard for describing entities / resources. Simply speaking, it is a way and means to express things. RDF is represented as SPO three tuple, sometimes called a sentence (statement). In knowledge map, we call it knowledge.



Figure 3. Structure of RDF

The Web Ontology Language (OWL) is a semantic Web language, which is used to express rich and complex knowledge about things, groups of things and relations between objects. OWL is a language based on computational logic, so the knowledge represented in OWL can be used by computer programs, for example, OWL documents, called ontology, can be published on the world wide web, and can be referenced or referenced from another OWL ontology. /12/

2.4 Tools used to build a Knowledge Graph

2.4.1 Protégé

Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies.

2.4.2 D2RQ

The D2RQ Platform is a system for accessing relational databases as virtual, read-only RDF graphs. It offers RDF-based access to the content of relational databases without having to replicate it into an RDF store. Using D2RQ you can:

- query a non-RDF database using SPARQL
- access the content of the database as Linked Data over the Web
- create custom dumps of the database in RDF formats for loading into an RDF store
- access information in a non-RDF database using the Apache Jena API

D2RQ is Open Source software and published under the Apache license.

2.4.3 Apache Jena

Apache Jena is an open source semantic web framework for java. Figure 4 illustrates the structure of a project using this framework. It provides APIs for extracting and writing data from RDF graphs. These diagrams are represented as abstract "models." Models can get data from files, databases, URLs, or a combination of them. /5/ In Apache Jena the users can configure their own inference rules or use the built-in OWL and RDFS reasoners.

Jena supports serialization of RDF graphs to:

- A relational database
- RDF/XML

- Turtle
- Notation 3

Fuseki is an HTTP interface to RDF data. It supports SPARQL for querying and updating. The project is a sub-project of Jena and is developed as servlet. Fuseki can also be run stand-alone server as it ships preconfigured with the Jetty web server.

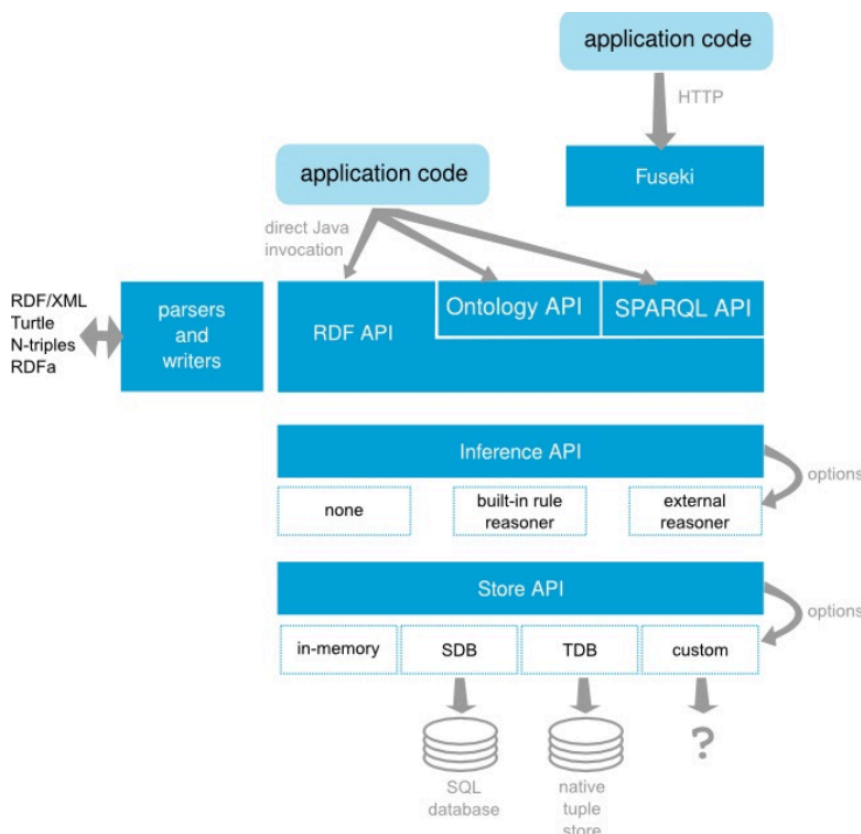


Figure 4. Apache Jena/5/

2.5 Natural Language Toolkit

Natural Language Toolkit is a leading platform for building Python programs to work with human language data. It was developed by Steven Bird and Edward Loper. /6/

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK not only be used as a teaching tool but also as an individual study tool. It is a platform for prototyping and building research systems. More than 32 universities in US and 25 countries using NLTK in their courses. NLTK supports

classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities. /9/

The process of classifying words into word classes and labeling them accordingly is called part-of-speech tagging, POS tagging, or simple tagging. Word classes are also referred to as part of speech or vocabulary. The set of tags for a particular task is called a set of tags.

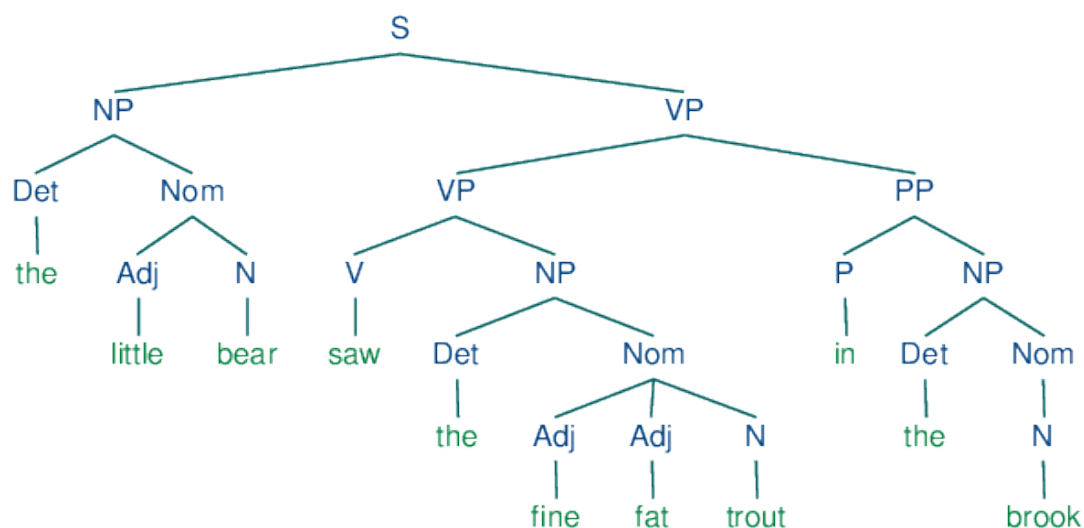


Figure 5. Parse tree generated with NLTK

2.6 Speech Recognition

Speech recognition is a cross-disciplinary field in computational linguistics, developing methods and techniques to enable computers to recognize and translate spoken language into text. It is also known as "automatic speech recognition" (ASR), "computer speech recognition", or just "voice to text" (STT). It combines knowledge and research in the field of linguistics, computer science, and electronic engineering.

This technology has a long history and has gone through several waves of innovation. Recently, this technology has deeply benefited from deep learning and big data. These advancements not only reflect the explosion of academic papers published in this field but more importantly, various industries around the world have adopted various deep learning methods when designing and deploying speech recognition systems.

Nowadays many companies provide their own speech recognition API for developers to use in their own application such as Bing, Google, Microsoft and Google. However, compared with other API, Google Cloud Speech API is performed better. /8/ So, in this thesis Google Cloud Speech API is used to convert speech to text.

File	Sphinx4		Google API		Microsoft API	
	WA	WER	WA	WER	WA	WER
TSX223	0.88	0.25	1.0	0.0	0.88	0.13
TSX293	0.64	0.36	0.82	0.18	1.0	0.0
TSI1894	0.78	0.22	1.0	0.0	0.78	0.22
TSI1400	0.79	0.29	0.93	0.07	1.0	0.0
TSX188	0.33	0.67	1.0	0.0	0.0	0.1
TSI1628	0.42	0.58	0.83	0.17	0.17	0.83
TSX314	0.67	0.33	1.0	0.0	1.0	0.0
DIG001	0.93	0.07	1.0	0.0	1.0	0.0
TSX216	0.89	0.11	1.0	0.0	1.0	0.0
TSX209	0.71	0.29	1.0	0.0	0.71	0.29
TSI1584	0.38	0.62	0.46	0.54	0.46	0.54
TSX371	0.45	0.55	1.0	0.0	0.91	0.09
TSI1373	0.29	0.71	1.0	0.0	0.57	0.43
TSX233	0.71	0.43	0.71	0.14	0.71	0.29
OSE003	0.63	0.5	0.63	0.38	0.63	0.38
AENGM8	0.89	0.11	1.0	0.0	1.0	0.0
AENGF8	0.33	0.67	0.78	0.22	1.0	0.0
AENGF7	0.83	0.17	1.0	0.0	1.0	0.0
AENGM2	0.86	0.14	1.0	0.0	0.86	0.14
Mean	WER: 0.37		WER: 0.09		WER: 0.18	

Figure 6. Speech Recognition Comparison /8/

2.7 NAO Robot

2.7.1 Introduction to Nao Robot

The humanoid robot Nao is developed by a French company named Aldebaran Robotics, which was acquired by SoftBank Group in 2015 and rebranded as SoftBank Robotics. The Nao robot is widely used in academic institutions. By the end of 2014, more than 5,000 Nao robots were in use with educational and research institutions in 70 countries /10/. Nao can be used as a research robot in schools, universities and universities. It is responsible for teaching programming and developing human-machine interaction/11/. Figure 6 shows the construction and functions of the Nao Robot

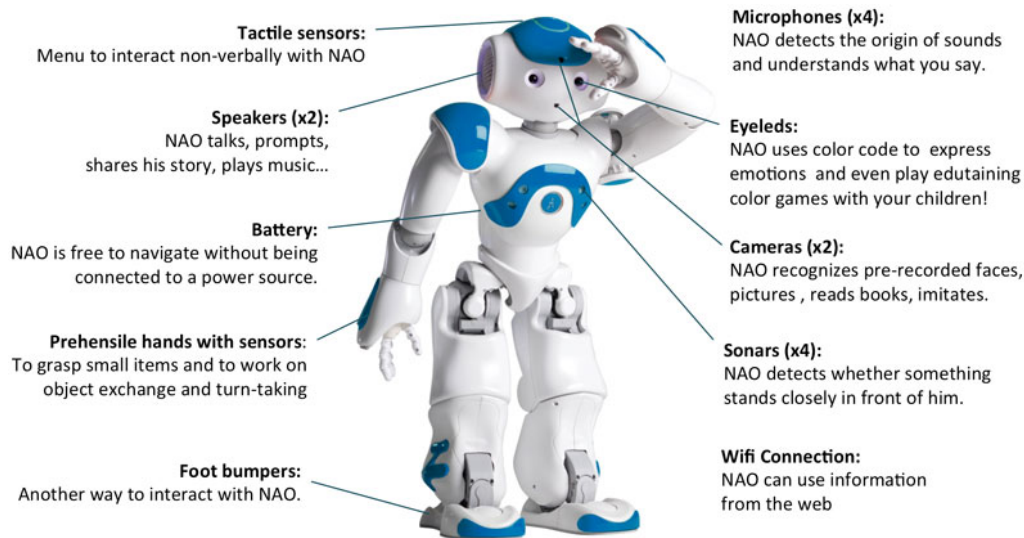


Figure 7. Nao Robot /14/

It has 25 degrees of freedom and human shape, it can move and adapt to the surrounding environment. Many sensors are used in several parts of the body, such as head, head and foot, so that robots can do this. In addition, Nao's vision comes from two cameras, capturing environment and helping Nao identify projects. In addition, Nao can connect to the Internet through different ways of connection (Wi-Fi and Ethernet). Specifications of the robot is shown in Figure 8.

Nao V5 Evolution (2014) ^{[4][24]}	
Height	58 centimetres (23 in)
Weight	4.3 kilograms (9.5 lb)
Power supply	lithium battery providing 48.6 Wh
Autonomy	90 minutes (active use)
Degrees of freedom	25
CPU	Intel Atom @ 1.6 GHz
Built-in OS	NAOqi 2.0 (Linux-based)
Compatible OS	Windows, Mac OS, Linux
Programming languages	C++, Python, Java, MATLAB, Urbi, C, .Net
Sensors	Two HD cameras, four microphones, sonar rangefinder, two infrared emitters and receivers, inertial board, nine tactile sensors, eight pressure sensors
Connectivity	Ethernet, Wi-Fi

Figure 8. Specifications of NAO/15/

2.7.2 NAOqi

NAOqi is a main software used to run on the robot and control it. The NAOqi framework is the programming framework used to program NAO. The framework allows uniform communication between different modules (motion, audio and video), isomorphic programming and uniform information sharing.

NAOqi is cross-platform. This framework is able to develop on Windows, Mac or Linux. And it also cross-language, which means it has an identical API for both C++ and Python. It gives developer a lot room to program with.

Figure 9 shows the process of NAOqi. The NAOqi executable running on the robot is a broker. When it starts, it loads a preference file named `autoload.ini`, which defines the library that it should load. Each library contains one or more modules that use agents to propagate its methods. The agent provides lookup service so that any module in the tree or in the whole network can find any published methods. The loading module forms the method tree attached to the module and the module attached to the agent.

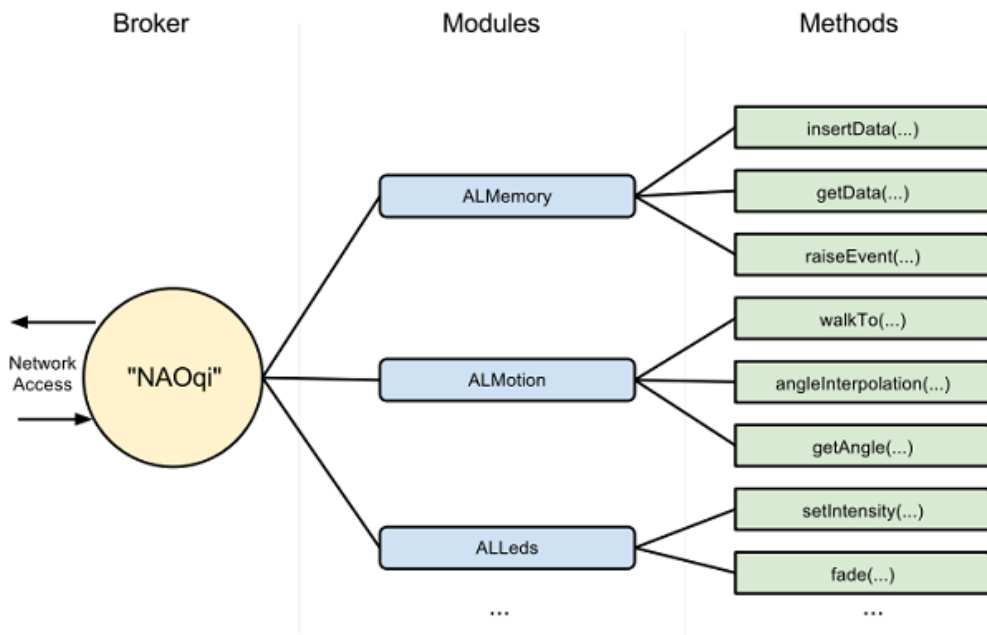


Figure 9. The NAOqi process/17/

3 OVERALL DESIGN

3.1 Structure of the whole project

The whole structure of the system is shown in Figure 10. First of all, by using the web crawler technology, the information of the hospital was gathered from their official websites such as department name, location, telephone number and open time. These data were stored into the database. Secondly, based on the relationship between these data, ontology was defined, combined with which the data was converted to RDF type. Then RDF is deployed on Jena, the data can be queried by SPARQL language.

When the user asks a question to Nao robot, the question will be recorded as a wav file and be uploaded to the Google Cloud Speech API to recognize, after which the question text will be returned. By analyzing the question text, the keywords will be extracted and the natural language will be transferred into SPARQL language. Then through Apache Jena SPARQL endpoint, the answer can be queried from the RDF. Finally, Nao robot can speak the answer by using its TTS module.

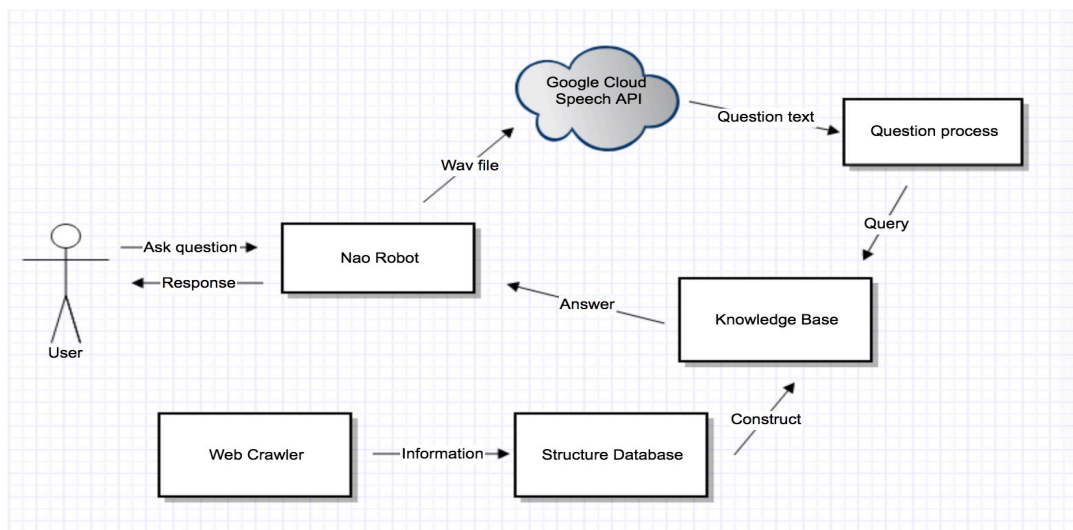


Figure 10. Whole structure

3.2 Flowchart of the whole project

As shown in Figure 11, the user asks question and the voice file is sent for speech recognition. Then the text of the question will be processed to search in the knowledge base to get the answer. Finally, speech is synthesized to answer the question.

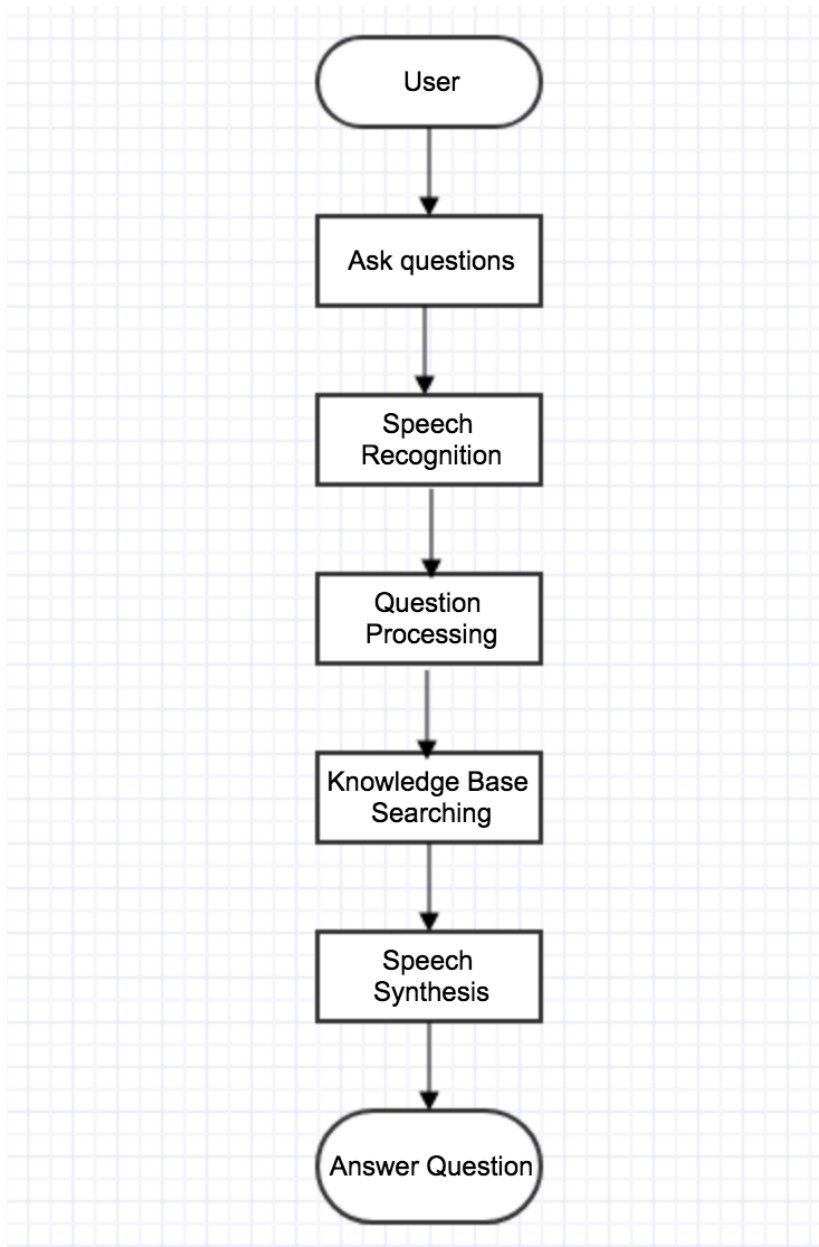


Figure 11. Flowchart

4 WEB INFORMATION EXTRACTION

4.1 Web Crawler

Web information extraction starts from a web crawler, which can also be called Spider or Robot. The Internet includes endless web information and this information is stored on the server over the word. Users go through the different websites and find the content that they are interested in by a hyperlink. The web crawler can download information from the website automatically. In order to gather the information that used as the answer in this system, the web crawler is used to crawl data from the website. However different websites have a different structure and some even have the different language, for example, some website contains not only English but also Finnish, which led to some problem for the web crawler.

4.1.1 Preparation

In this thesis, we mainly analyze the official website of hospitals in Finland. The contact information of hospitals in Helsinki and detail information about outpatient clinics and wards of Vaasa center hospital, Turku university hospital (main hospital), and Tampere university hospital (central hospital) was gathered for the system. Below is a list of the websites.

- <https://www.vaasankeskussairaala.fi/en/>
- <https://www.pshp.fi/>
- <http://www.vsshp.fi/en/toimipaikat/tyks/osastot-ja-poliklinikat/Pages/default.aspx>

4.1.2 Libraries

Python is the programming language used in this project, and there are many libraries in that can be used to achieve our purpose.

- Requests
Requests is the only Non-GMO HTTP library for Python. Requests allows user to send organic, grass-fed HTTP/1.1 requests without manually add query strings to URLs, or from-encode POST data.

- BeautifulSoup

BeautifulSoup is a Python library that can extract data from HTML or XML files. It can be used to navigate, find, and modify documents by your favourite converter. BeautifulSoup not only supports HTML parser, but also supports some third-party parser, such as lxml, XML, html5lib, but needs to install corresponding library.

- Regular expression

A regular expression (or RE) specifies a set of strings that match it; the function in this module allows you to check whether a particular string matches a given regular expression (or whether a given regular expression matches a particular string, the string can be attributed to the same thing). It can be used for screening information.

4.2 Information Extraction

According to the content above, the related information can be extracted from the HTML file which was crawled from websites. Figure 12 is an example of a website. In this chapter, we analyze the methods to extract information from a HTML file.

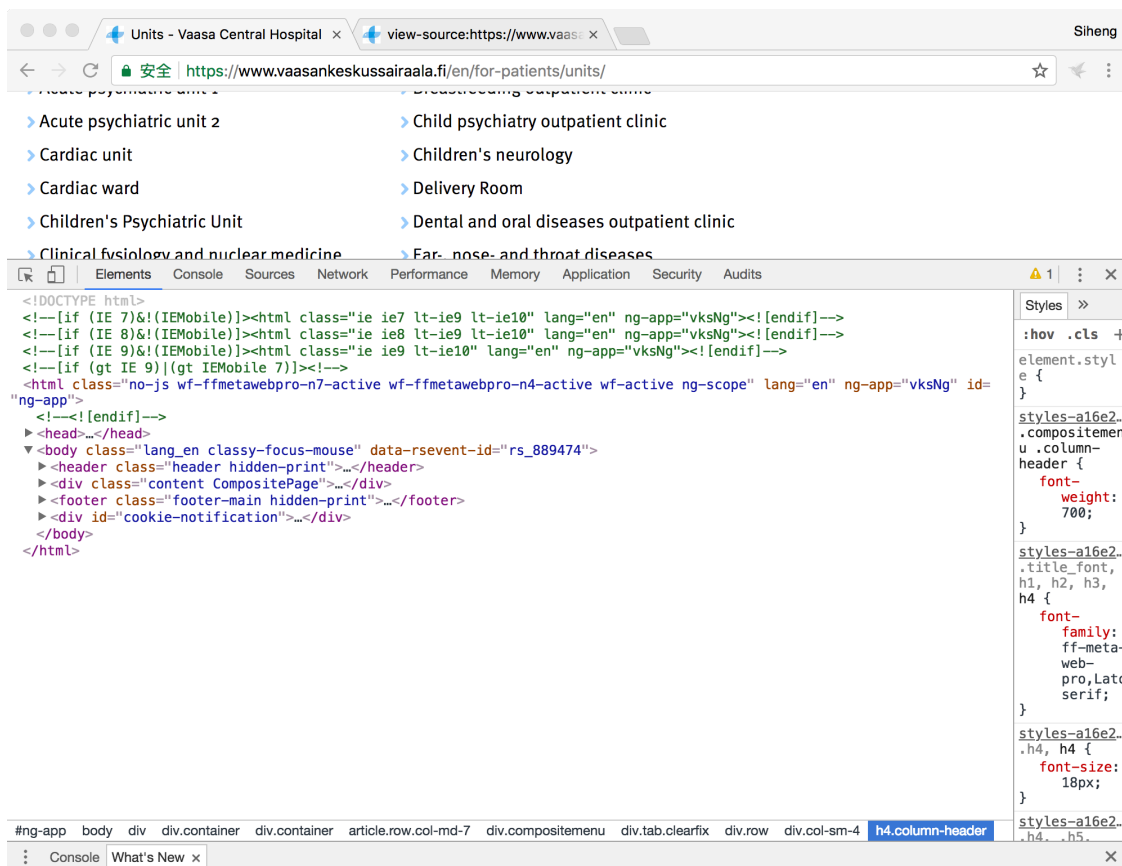


Figure 12. HTML Example of Website

As shown in Figure 12, an HTML page mainly contain two parts, head part and body part. Information is usually written in the body part with relative tags. Figure 13 lists the common tags that the developer always uses in HTML.

Tag	Description
<code><html> ... </html></code>	Declares the Web page to be written in HTML
<code><head> ... </head></code>	Delimits the page's head
<code><title> ... </title></code>	Defines the title (not displayed on the page)
<code><body> ... </body></code>	Delimits the page's body
<code><h <i>n</i>> ... </h<i>n</i>></code>	Delimits a level <i>n</i> heading
<code> ... </code>	Set ... in boldface
<code><i> ... </i></code>	Set ... in italics
<code><center> ... </center></code>	Center ... on the page horizontally
<code> ... </code>	Brackets an unordered (bulleted) list
<code> ... </code>	Brackets a numbered list
<code> ... </code>	Brackets an item in an ordered or numbered list
<code>
</code>	Forces a line break here
<code><p></code>	Starts a paragraph
<code><hr></code>	Inserts a horizontal rule
<code></code>	Displays an image here
<code> ... </code>	Defines a hyperlink

Figure 13. HTML Element List

Figure 14 is a part of the HTML of the website. It can be seen that information is written in the middle of the HTML tags. In order to get the needed information, the whole webpage information should be divided into corresponding blocks according to the logic of the content, each part is a separate theme. It is the HTML DOM tree. HTML DOM tree is a normal structure of a HTML document, it is like a tree which has a lot of leaves and the information is written layer by layer.

```

▼ <div class="article-body col-md-8">
  ▶ <p>...</p>
  ▶ <p>...</p>
  ▶ <p>...</p>
  ▶ <p>...</p>
  ▶ <p>...</p>
  ▼ <div class="contact-box visible-xs visible-sm">
    ▼ <div class="contact">
      <h4>Contact info</h4>
      <h5>Location</h5>
      <div>B4</div>
      <h5>Clinic hours</h5>
      ▶ <div>...</div>
      <h5>Kanslia / Kansli</h5>
      <div>06 213 2032</div>
      <div></div>
    </div>
  </div>
  ▶ <footer>...</footer>
</div>

```

Figure 14. HTML

BeautifulSoup 4 in Python is used for pulling data out of HTML files. It provides some methods to find the specific tag by its attributes like class name. Firstly, from the seed URL, the home page is retrieved and a list of department URLs were found as hyperlinks and they will be stored in a list. As it is shown in Figure 15, the HTML DOM Tree of the sub-websites and HTML structure are almost the same, it is easy extract the data in a loop.

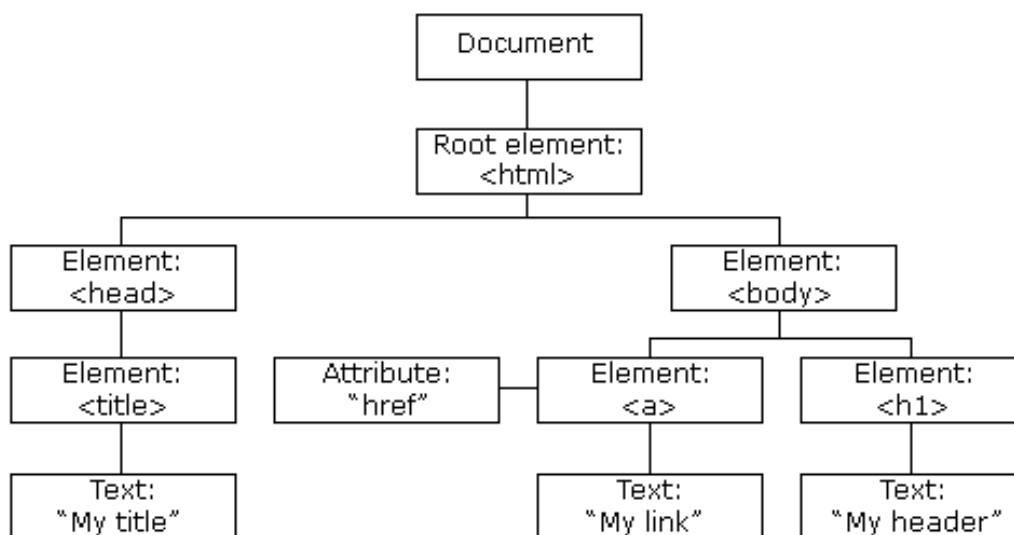


Figure 15. HTML DOM Tree

Breastfeeding outpatient clinic



Figure 16. Information on the Website

Below it shows the example code of how to extract data from the website. However, in some websites, the HTML DOM Tree is not so normative, which makes it difficult to parse the HTML file with BeautifulSoup by an finding element through the element class name. In this situation, the regular expression is used to match the string and get the data we needed. For example, in the sample HTML file shown below, the information is separated by the tag `<p>` without any attributes. In order to solve this problem, RE is used to match the content of `<h3>` and according to the text in `<h3>` get the data respectively.

```
content = requests.get(url).content
soup1 = BeautifulSoup(content, 'html.parser', from_encoding='utf-8')
content1 = soup1.find(name='div', attrs={"class": "contact-box hidden-xs hidden-sm"})
content2 = content1.find(name='div', attrs={"class": "contact"})
content3 = content2.find_all(name='div')
location.append(content3[0].get_text())
time.append(content3[1].get_text())
phone.append(content3[2].get_text())
```

Home > Units > Tyks Main Hospital, Turku > Outpatient Clinics and Wards > Acute Care Surgery Ward

Outpatient clinics and wards
Emergency Medical Services
Halikko Hospital
Psychiatric Units
Public Utility Tyks-Sapa
Tyks Main Hospital, Turku

ACUTE CARE SURGERY WARD

Contact information

Street address
Tyks T-hospital
Building 18
Main entrance 18A
3rd floor, G Wing
Hämeentie 11, Turku

Telephone
02 313 7243

```

<div id="uetarpagesstatusbar"></div>
<div id="DeltaPlaceholderMain">
  <a id="mainContent" name="mainContent" tabindex="-1"></a>
  <div class="article article-content yksikko">
    <div class="right-zone">
      <h1 class="title-mob"> Acute Care Surgery Ward</h1>
      <div class="mini-content">
        <div id="ctl00_PlaceHolderMain_ctl01_label" style="display:none">Page Content</div>
        <div id="ctl00_PlaceHolderMain_ctl01_ControlWrapper_RichHtmlField" class="ms-rtestate-field" style="display:inline" aria-labelledby="
          <h2>Contact information</h2> = s0
          <h3>Street address</h3>
          <p></p>
          <h3>Telephone</h3>
          <p>02 313 7243</p>
          <h3>Fax</h3>
          <p>02 313 7244</p>
          <h3>Postal address</h3>
          <p>PO Box 52, 20521 Turku</p>
        </div>
        <menu class="ms-hide"></menu>
      </div>
    </div>
  </div>

```

Figure 17. Sample HTML

4.3 Insert into Database

By using BeautifulSoup and Regular expression, we get the data which is the answer of our system. The next step is to store this data in the database for the next usage. MySQL is the database used in this project and MySQL Workbench is a unified visual tool for manage the database.

Pymysql is the library used to connect the database from python application. “Pip install pymysql” is the command used to install this package in the terminal. The code below is about how to create tables in the database and insert data into the table.

Import pymysql

```

conn = pymysql.Connect(
    host='localhost',
    port=3306,
    user='root',
    password='****',
    db='****',
    charset='utf8'

```

```

)

try:
    with conn.cursor() as cur:

        create_sql = """
        CREATE TABLE Department
        (department_id INT(11) PRIMARY KEY NOT NULL,
        department_name VARCHAR(100),
        department_location VARCHAR(45),
        department_time VARCHAR(200),
        department_phone VARCHAR(45))
        """

        insert_sql = """
        INSERT INTO Department(department_id, department_name, department_location,
        department_time, department_phone)
        VALUES('%s', '%s', '%s', '%s', '%s' )"""
        cur.execute(create_sql)
        for i in range(0, len(dName)-1):
            insert_data = (i, dName[i-1], location[i-1], time[i-1], phone[i-1])

            cur.execute(insert_sql%tuple(insert_data))

        conn.commit()

finally:
    conn.close()

```

Figure 18 is part of the screenshot of the database. The unit name, location, open time and telephone number have been gathered as the answer.

department_id	department_name	department_location	department_time	department_phone
0	surgery	T1	Arkisin klo 8.00-15.00 /Vardagar kl. 8.00-15.00	06 213 1522
1	breastfeeding outpatient clinic	A4	Ma ja to klo 8.30-15.00 /Mån. och tors. kl. 8.30-...	044 323 2072
2	child psychiatry outpatient clinic	H1	Arkisin klo 8.00-16.00 / Vardagar kl. 8.00-16.00	06 213 2241
3	delivery room	B4	Ympäri vuorokauden /Dygnet runt	06 213 2032
4	dental and oral diseases outpatient clinic	A7	Ma-to klo 7.30-15.30 ja pe klo 7.30-15.00 / Mån.-...	06 213 1572
5	ear-, nose- and throat diseases	A7	Arkisin klo 8.00-15.00 /Vardagar kl. 8.00-15.00	06 213 1372
6	eating disorders outpatient clinic	RC2	Ma-to klo 8.00-15.30 ja pe klo 8.00-14.00 /Mån.-t...	06 213 2130
7	eye disease outpatient clinic	A6	Arkisin klo 7.00-15.00 /Vardagar kl. 7.00-15.00	06 213 1592
8	general hospital psychiatry outpatient clinic	A1	Arkisin klo 8.30-15.30 /Vardagar kl. 8.30-15.30	06 213 2251
9	gynaecologic outpatient clinic	A5	Ma-to klo 7.30-15.30 ja pe klo 7.30-14.30 /Mån.-t...	06 213 2012
10	hospital chaplains	H4, C0		06 213 1084
11	internal medicine outpatient clinic	D1	Arkisin klo 10.00-12.00 /Vardagar kl. 10.00-12.0...	06 213 2632
12	lung diseases	U3	Arkisin klo 8.00-15.00 /Vardagar kl. 8.00-15.00	06 213 2997
13	maternity outpatient clinic	A8	Arkisin klo 8.00-15.00 /Vardagar kl. 8.00-15.00	06 213 2022
14	oncology outpatient clinic	Q1	Arkisin klo 8.00-15.00 /Vardagar kl. 8.00-15.00	06 213 2910
15	paediatric outpatient clinic	U2	Arkisin klo 8.00-15.00 / Vardagar kl. 8.00-15.00	06 213 1922
16	pain management	Y2	Puhelinaika pe klo 12.00-14.30	06 213 1321
17	physical rehabilitation guidance	H6	Arkisin klo 8.00-16.00 / Vardagar kl. 8.00-16.00	06 213 2412
18	presurgical clinic	Y2	Ma - to klo 7.00 - 15.00 ja pe klo 7.00 - 14.00 /M...	06 213 4486

Figure 18. Database

5 KNOWLEDGE BASE CONSTRUCTION

According to the data about hospital that we get from their official website, some data about diseases were added manually to the database in order to make the structure of the knowledge base complete.

There are totally three tables, one table about departments of the hospital and one table containing information about disease. They are then combined with each other by the unique id in the third table Disease to Department. Figure 19 is the ER figure of the database.

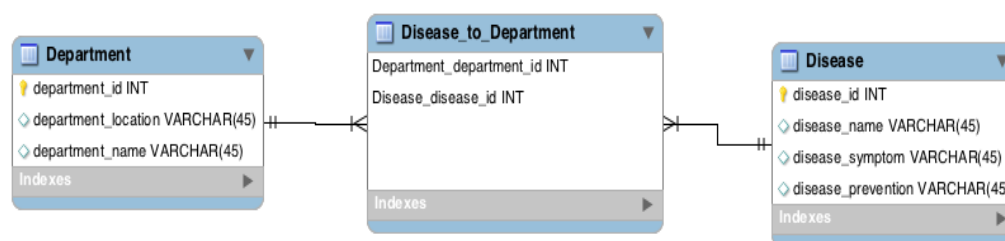


Figure 19. ER Figure of Database

5.1 Ontology modeling

There are basically two ways to build an ontology: top-down and bottom-up.

The ontology construction of knowledge base in an open domain usually adopts the bottom-up approach to extract concepts, concept hierarchy, and concept relations automatically from knowledge map. It is also well understood that the open domain is too complicated to be considered in a top-down way, and the corresponding concept is growing as the world changes.

Limited domain knowledge maps mostly use the top-down method to build an ontology. On the one hand, the concept and scope of domain knowledge maps are fixed or controllable relative to the open domain knowledge map; on the other hand, we require higher accuracy for the domain knowledge atlas. Most of the knowledge atlas behind some of the voice aides we have come into contact with is domain knowledge maps, such as music knowledge atlas, sports knowledge atlas, cooking knowledge atlas, and so on. It is precisely because of these domains that knowledge maps satisfy most of the needs of users, but also need to ensure their accuracy. This example is a knowledge map in the medical

field. We use the top-down approach to build ontology structure. Protégé is the tool used to build ontology.

1. Open protégé, the interface is shown below. Fill in the IRI of our new ontology resource in Ontology IRI in the red frame.

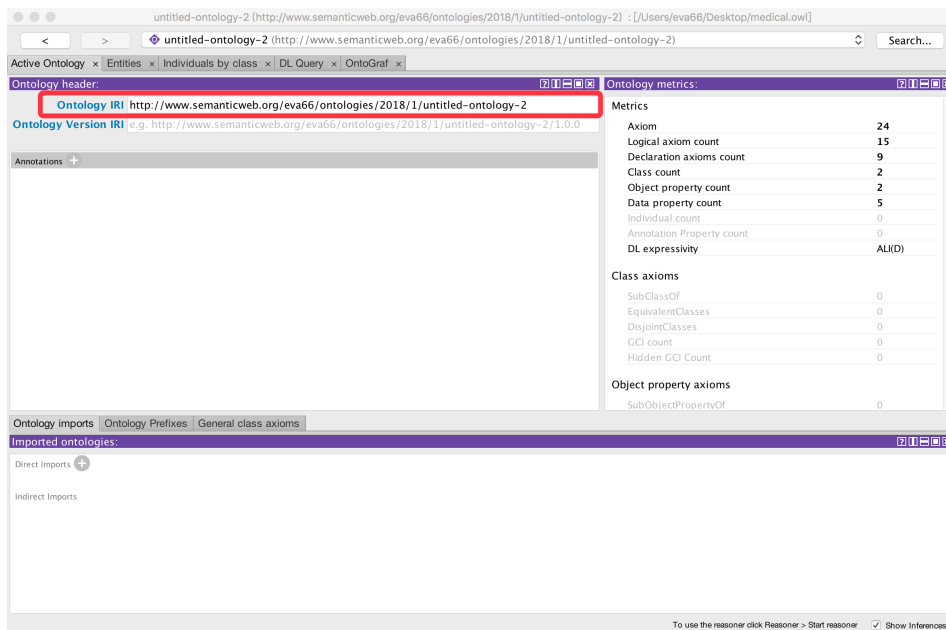


Figure 20. IRI

2. In the “Entities” tab, “Classes” is chosen. In this page, the classes of this knowledge base are created and there are all the sub-class of “Thing”. As shown in Figure 21, department class and disease class are defined.

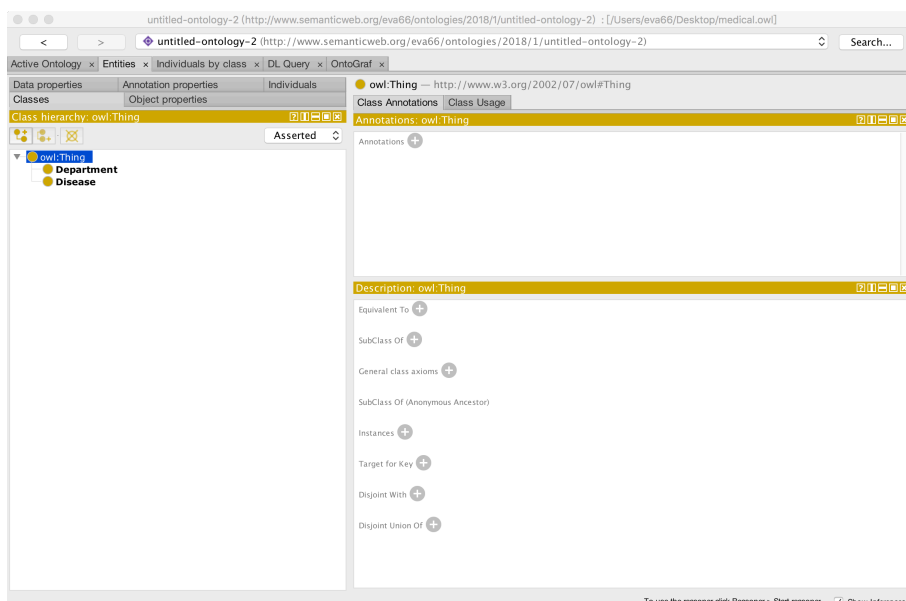


Figure 21. Classes

3. In “Object properties” tag, the relationship between the classes was created, which means the object property. Here two relationships were defined, “hasBelong” means the disease belong to a department and its domain is disease and rang is department. This relationship is inverse of “hasDiseas”.

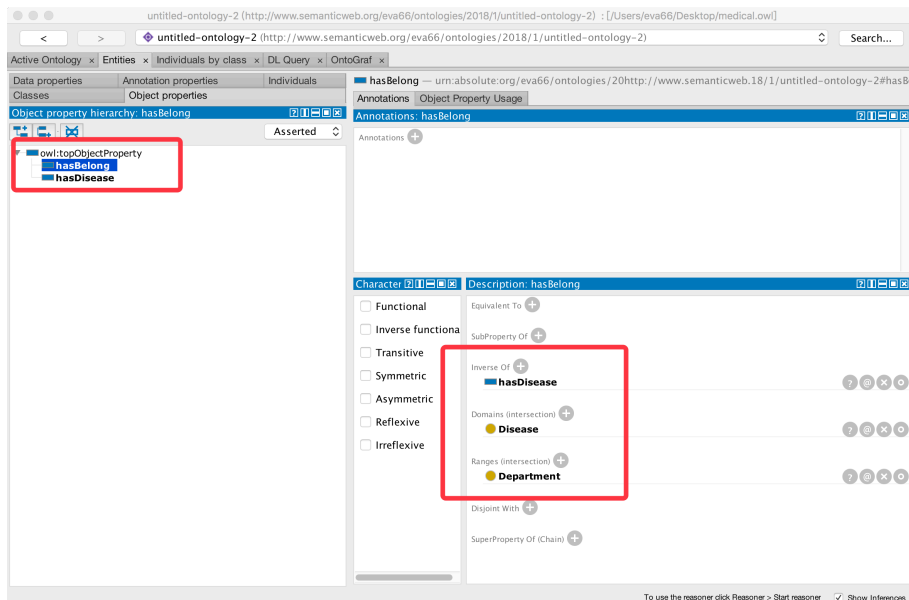


Figure 22. Object Property

4. Finally, in “Data properties” tag in this interface the data attributes were defined. The data attribute is equivalent to the leaf node of the tree, and it only goes in, but not out.

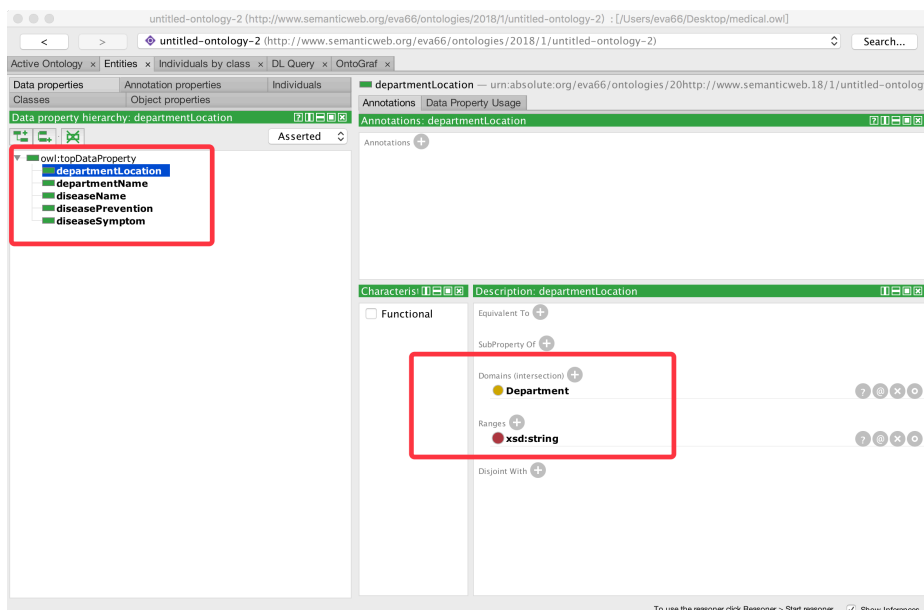


Figure 23. Data Properties

Protégé also supports the visualization of ontology structure. Clicking on the "Window" option, select "OntoGraf" in "Tabs". Moving elements in the right window one can directly observe the relationship between ontologies.

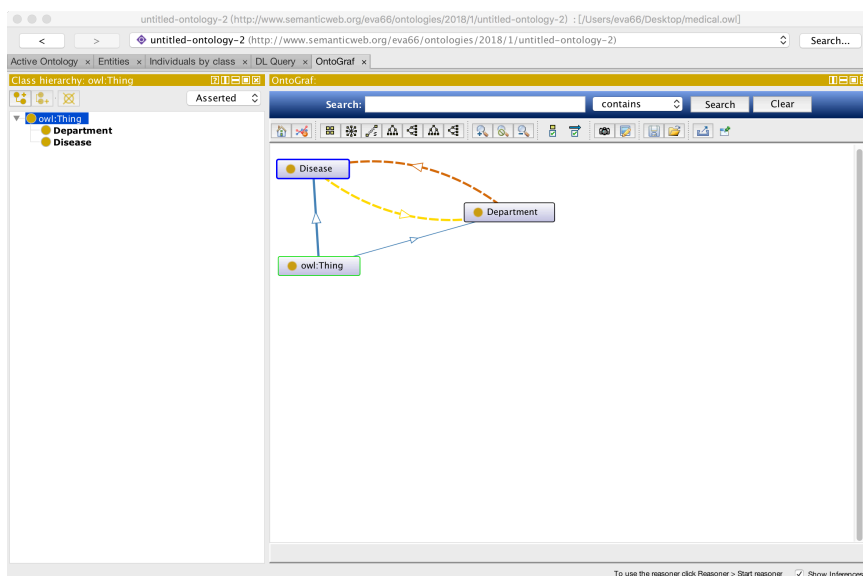


Figure 24. OntoGraf

5.2 Relational database to RDF

D2RQ can access the database as a virtual RDF Graph. Its mechanism is to translate RDF queries and other operations into SQL statements through mapping files, and finally achieve corresponding operations on RDB. When we do a knowledge map project, we can flexibly choose the way of data access. When external services are provided and query operations are frequent, it is better to directly convert RDB data to RDF, which will save a lot of time to convert SPARQL to SQL.

D2RQ provides its own mapping language, which is similar to R2RML. D2RQ has released r2rml-kit to support the two mapping standards formulated by W3C. D2RQ has a more convenient place to automatically generate a predefined mapping file based on your database, which users can modify on this file to map the data to their own ontology. In this project, data relationship is relatively simple, editing R2RML files or modifying efficiency on D2RQ generated mapping files. When data relations are very complex, it saves a lot of time to modify directly on the mapping files generated by D2RQ. D2RQ's mapping language is also very concise, and also supports mapping the SQL results, whose

SQL is expressed implicitly with condition keywords, unlike R2RML as an explicit SQL statement.

Download D2RQ, enter its directory in terminal, run the following command to generate the default mapping file:

```
generate-mapping -u root -p pass -o mapping.ttl jdbc:mysql:///medical
```

Root is the username of MySQL, pass is the password of the database, "jdbc:mysql:///medical" is the database we want to convert. Next, change the default mapping vocabulary into the words in our ontology. When dealing with foreign keys, we should pay attention to the domain and range of the current editing properties, belongsToClassMap is the domain and refersToClassMap is the range.

Use the following command to transform data to RDF:

```
Dump-rdf -o mapping.nt mapping.ttl
```

Mapping.ttl is the modified mapping file. It supports the exported RDF format with "TURTLE", "RDF/XML", "RDF/XML-ABBREV", "N3" and "N-TRIPLE". "N-TRIPLE" is the default output format. Below showing a part of data in the N-TRIPLE file.

```
<file:///Users/eva66/Downloads/d2rq-0.8.1/QA.nt#Department/0> <http://
www.semanticweb.org/eva66/ontologies/2018/2/medical#departmentName>
"surgery" .
<file:///Users/eva66/Downloads/d2rq-0.8.1/QA.nt#Department/0> <http://
www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.semanticweb.org/
eva66/ontologies/2018/2/medical#Department> .
<file:///Users/eva66/Downloads/d2rq-0.8.1/QA.nt#Department/1> <http://
www.semanticweb.org/eva66/ontologies/2018/2/medical#departmentPhone>
"044 323 2072" .
<file:///Users/eva66/Downloads/d2rq-0.8.1/QA.nt#Department/1> <http://
www.semanticweb.org/eva66/ontologies/2018/2/medical#departmentTime> "Ma
ja to klo 8.30-15.00 /M\u00E5n. och tors. kl. 8.30-15.00" .
<file:///Users/eva66/Downloads/d2rq-0.8.1/QA.nt#Department/1> <http://
www.semanticweb.org/eva66/ontologies/2018/2/medical#departmentLocation>
"A4" .
<file:///Users/eva66/Downloads/d2rq-0.8.1/QA.nt#Department/1> <http://
www.semanticweb.org/eva66/ontologies/2018/2/medical#departmentName>
"breastfeeding outpatient clinic" .
<file:///Users/eva66/Downloads/d2rq-0.8.1/QA.nt#Department/1> <http://
www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.semanticweb.org/
eva66/ontologies/2018/2/medical#Department> .
<file:///Users/eva66/Downloads/d2rq-0.8.1/QA.nt#Department/2> <http://
www.semanticweb.org/eva66/ontologies/2018/2/medical#departmentPhone> "06
213 2241" .
```

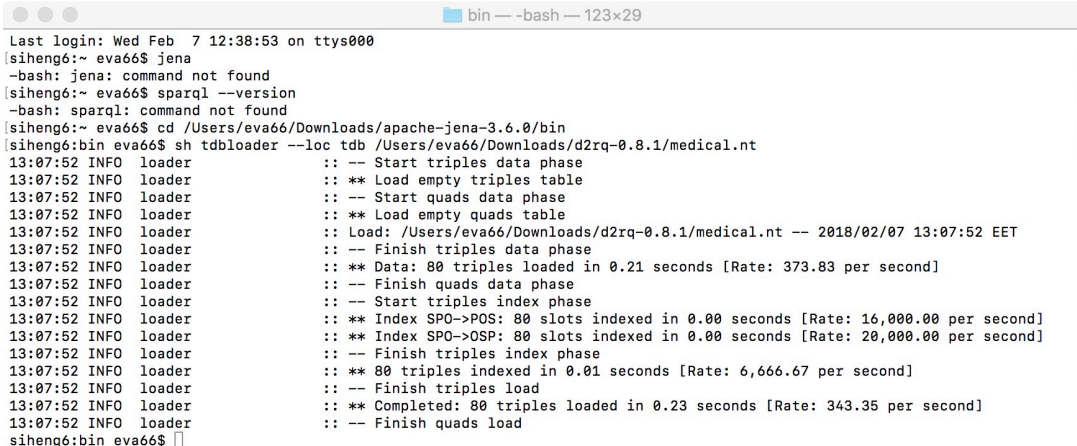
Figure 25. NT file

5.3 Apache Jena SPARQL endpoint and reasoning

In this part, there three components of Jena are used: TDB, rule reasoner and Fuseki.

1. TDB is Jena's component used to store RDF, and it belongs to the technology of storage level. In a single machine case, it can provide very high RDF storage performance. At present, the latest version of TDB is TDB2, and it is not compatible with TDB1.
2. Jena provides RDFS, OWL, and general rule reasoning machines. In fact, Jena's RDFS and OWL inference engine is also implemented by Jena's own general rule inference machine.
3. Fuseki is the SPARQL server provided by Jena, that is SPARQL endpoint. It provides four modes of operation: single machine operation, running as a service of the system, running as and web application, or running as an embedded server.

First of all, apache-jena and apache-jena-fuseki need to be download from their websites. Then, creating a directory to store TDB data. Entering the bat directory of the "apache-jena-X.X.X" folder, you can see a lot of batch files, and we use "tdbloader.bat" to store our RDF data in the TDB way. The command is as follows:



```

Last login: Wed Feb  7 12:38:53 on ttys000
siheng6:~ eva66$ jena
-bash: jena: command not found
siheng6:~ eva66$ sparql --version
-bash: sparql: command not found
siheng6:~ eva66$ cd /Users/eva66/Downloads/apache-jena-3.6.0/bin
siheng6:bin eva66$ sh tdbloader --loc tdb /Users/eva66/Downloads/d2rq-0.8.1/medical.nt
13:07:52 INFO loader      :: -- Start triples data phase
13:07:52 INFO loader      :: ** Load empty triples table
13:07:52 INFO loader      :: -- Start quads data phase
13:07:52 INFO loader      :: ** Load empty quads table
13:07:52 INFO loader      :: Load: /Users/eva66/Downloads/d2rq-0.8.1/medical.nt -- 2018/02/07 13:07:52 EET
13:07:52 INFO loader      :: -- Finish triples data phase
13:07:52 INFO loader      :: ** Data: 80 triples loaded in 0.21 seconds [Rate: 373.83 per second]
13:07:52 INFO loader      :: -- Finish quads data phase
13:07:52 INFO loader      :: -- Start triples index phase
13:07:52 INFO loader      :: ** Index SPO->POS: 80 slots indexed in 0.00 seconds [Rate: 16,000.00 per second]
13:07:52 INFO loader      :: ** Index SPO->OSP: 80 slots indexed in 0.00 seconds [Rate: 20,000.00 per second]
13:07:52 INFO loader      :: -- Finish triples index phase
13:07:52 INFO loader      :: ** 80 triples indexed in 0.01 seconds [Rate: 6,666.67 per second]
13:07:52 INFO loader      :: -- Finish triples load
13:07:52 INFO loader      :: ** Completed: 80 triples loaded in 0.23 seconds [Rate: 343.35 per second]
13:07:52 INFO loader      :: -- Finish quads load
siheng6:bin eva66$

```

Figure 26. RDF to TDB

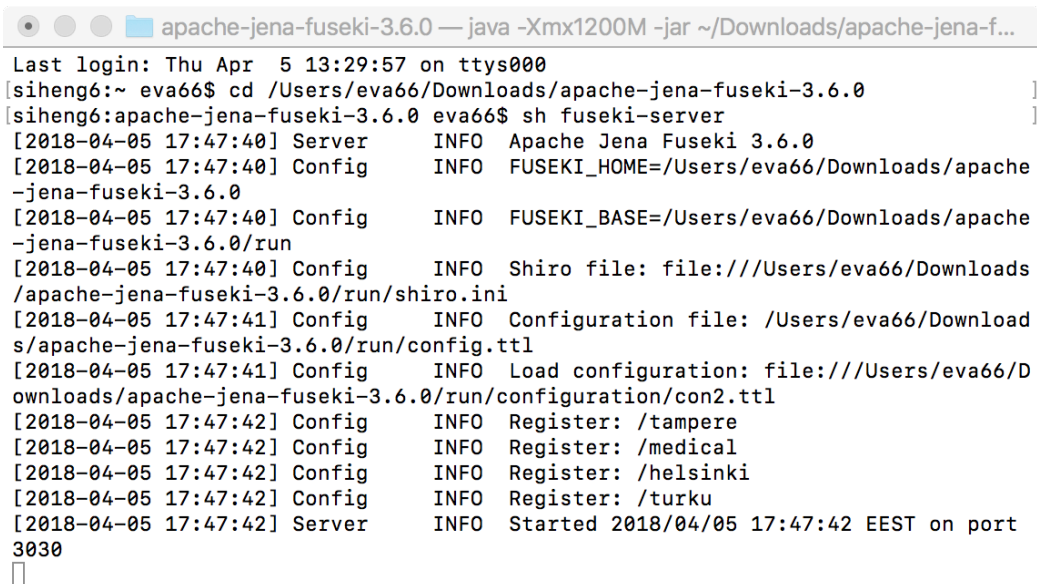
"--loc" specifies the location of TDB storage and the second parameter is RDF data transformed from MySQL data.

Since Jena provides general rule reasoning machines, it allows user to define their own rules. In Fuseki's folder, creating a new text file under the folder "databases" named "rules.ttl". OWL's reasoning function can also be implemented in a regular reasoning machine, so we define "ruleInverse" to represent the opposite relationship between "hasBelong" and "hasDisease".

Besides the rule file, the ontology.owl file which is exported from protégé is also needed to be added into the folder "databases" and change the ".owl" suffix name to ".ttl".

In the "run" folder under the "configuration", the configuration file should be created to mapping the TDB and rules. Part of this file can be checked in the appendix.

Then running "Fuseki-server" in the terminal, the result can be seen in Figure 27:



```

Last login: Thu Apr  5 13:29:57 on ttys000
[siheng6:~ eva66$ cd /Users/eva66/Downloads/apache-jena-fuseki-3.6.0 ]
[siheng6:apache-jena-fuseki-3.6.0 eva66$ sh fuseki-server ]
[2018-04-05 17:47:40] Server      INFO  Apache Jena Fuseki 3.6.0
[2018-04-05 17:47:40] Config      INFO  FUSEKI_HOME=/Users/eva66/Downloads/apache-jena-fuseki-3.6.0
[2018-04-05 17:47:40] Config      INFO  FUSEKI_BASE=/Users/eva66/Downloads/apache-jena-fuseki-3.6.0/run
[2018-04-05 17:47:40] Config      INFO  Shiro file: file:///Users/eva66/Downloads/apache-jena-fuseki-3.6.0/run/shiro.ini
[2018-04-05 17:47:41] Config      INFO  Configuration file: /Users/eva66/Downloads/apache-jena-fuseki-3.6.0/run/config.ttl
[2018-04-05 17:47:41] Config      INFO  Load configuration: file:///Users/eva66/Downloads/apache-jena-fuseki-3.6.0/run/configuration/con2.ttl
[2018-04-05 17:47:42] Config      INFO  Register: /tampere
[2018-04-05 17:47:42] Config      INFO  Register: /medical
[2018-04-05 17:47:42] Config      INFO  Register: /helsinki
[2018-04-05 17:47:42] Config      INFO  Register: /turku
[2018-04-05 17:47:42] Server      INFO  Started 2018/04/05 17:47:42 EEST on port 3030
□

```

Figure 27. Fuseki Sever

The default port of Fuseki is 3030, and the browser access "<http://localhost:3030/>".

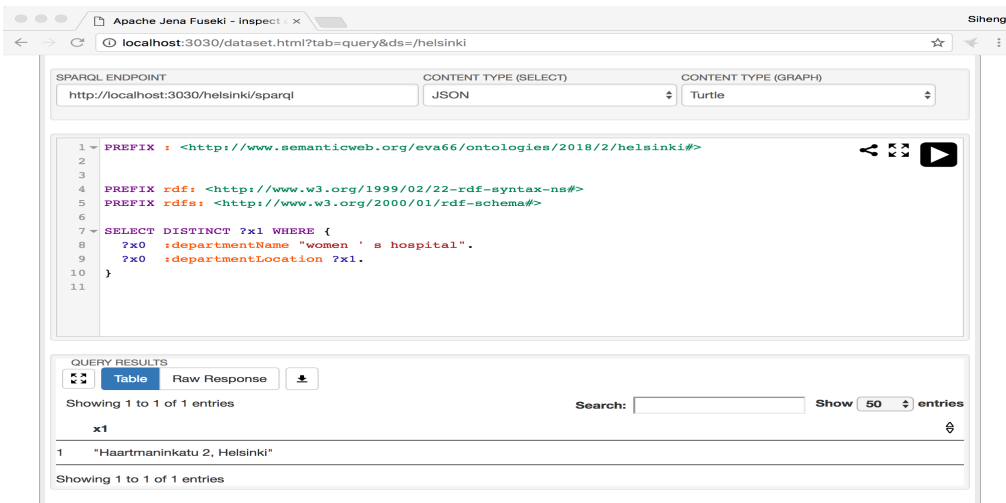


Figure 28. Browser access to Fuseki Sever

In Python, SPARQLWrapper is used to send query requests to the Fuseki server. The results are returned in JSON format. Figure 28 shows a part code of jena-sarql-endpoint about how to connect to the server and get result.

```

1  from SPARQLWrapper import SPARQLWrapper, JSON
2  from collections import OrderedDict
3
4
5  class JenaFuseki:
6      def __init__(self, endpoint_url):
7          self.sparql_conn = SPARQLWrapper(endpoint_url)
8
9      def get_sparql_result(self, query):
10         self.sparql_conn.setQuery(query)
11         self.sparql_conn.setReturnFormat(JSON)
12         return self.sparql_conn.query().convert()
13
14     @staticmethod
15     def parse_result(query_result):
16
17         try:
18             query_head = query_result['head']['vars']
19             query_results = list()
20             for r in query_result['results']['bindings']:
21                 temp_dict = OrderedDict()
22                 for h in query_head:
23                     temp_dict[h] = r[h]['value']
24                 query_results.append(temp_dict)
25             return query_head, query_results
26         except KeyError:
27             return None, query_result['boolean']
28

```

Figure 29. Connect Fuseki Sever

6 QUESTION TO SPARQL

Since the question is asked as in sentence of natural language, libraries are needed to complete the initial Natural Language Processing (participle, entity recognition), and then use the library which supports regular matching to complete the subsequent semantic matching.

Quepy is a Python framework to transform natural language questions into queries in a database query language. It can be easily customized to different kinds of questions in natural language and database queries. It can be installed with the command:

```
pip install quepy
```

6.1 Natural Language Parsing

NLTK is the library used to process the question sentence which provides methods to tag words in a text. After translating natural language into a word based basic unit, we use REFO (Regular Expressions for Objects) to complete the semantic matching. According to the specific tag of words, it will be matched with regex of the question template to extract the entity which is needed.

For example, the question “where is the presurgical clinic?” can be tagged as the following, and “presurgical clinic” is the entity and “where” is the relationship that needs to be queried.

```
(S where/WRB is/VBZ presurgical/JJ clinic/NN)
```

Since most of the entities that we want to get are nouns and adjectives, it can be expressed as the group of words with specific pos. As the code shown in figure 30, “thing” is the entity that is extracted from the sentence. And “regex” is the template of the question which should be matched. However, one kind of question may have several question templates to match and it makes the system more flexible.

```

thing = Group(Star(Pos("VBG") | Pos("POS") | Pos("JJ") | Pos("NNP") |
                  Pos("CC") | Pos("IN") | Pos("NN") | Pos("NP") |
                  Pos("NS") | Pos("VBP") | Pos("NNP") | Pos("NNPS")), "thing")

regex1 = Lemma("where") + Lemma("be") + \
         Question(Pos("DT")) + thing + Question(Pos("."))
regex2 = Question(Lemmas("the location")) + Lemma("of") + Question(Pos("DT")) + thing + Question(Pos("."))
regex = regex1 | regex2

```

Figure 30. Semantic matching

This regular expression matches questions of the form “where is X?”, but also “where was X?”, “where were X?” and other variants of the verb to be because it is using the lemma of the verb in the regular expression.

In this system, there are totally seven type of question that are pre-defined.

1. Where is (hospital/department)?
2. The opening time of (hospital/department)? / When is (hospital/department) open?
3. How to contact (hospital/department)? / The phone number of (hospital/department)?
4. The symptom of (disease)?
5. How to prevent (disease)?
6. Department of (disease)?
7. Diseases in (department)?

6.2 Query Generation & DSL

In quepy framework, the domain is mainly defined in the DSL file. Quepy uses an abstract semantics as a language-independent representation that is then mapped to a query language. This allows questions to be mapped to different query languages in a transparent manner. In Figure 31, it is an example of domain specific language for the information of hospital department.

```

class DepartmentName(HasKeyword):
    relation = ":departmentName"

class NameOfDepartment(FixedRelation):
    relation = ":departmentName"
    reverse = True

class LocationOf(FixedRelation):
    relation = ":departmentLocation"
    reverse = True

class TimeOf(FixedRelation):
    relation = ":departmentTime"
    reverse = True

class PhoneOf(FixedRelation):
    relation = ":departmentPhone"
    reverse = True

```

Figure 31. Domain Specific Language

After defining the domain specific language of the system, if a regex has a successful match with an input question, the interpret method will be called with the match to generate the query. Below is shown the interpret method for the example question “Where is XXX?”.

```

def interpret(self, match):
    thing = DepartmentName(match.thing.tokens)
    location = LocationOf(thing)
    return location

```

Figure 32. Interpret Method

In the main.py file there are some lines of code to use this application and the generated query.

```

import quepy

dbpedia = quepy.install("dbpedia")

target, query, metadata = dbpedia.get_query("where is a blowtorch?")

print query

```

PREFIX: <<http://www.semanticweb.org/eva66/ontologies/2018/2/helsinki#>>

PREFIX rdf: <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>>

PREFIX rdfs: <<http://www.w3.org/2000/01/rdf-schema#>>

```
SELECT DISTINCT? x1 WHERE {
```

```
  ? x0: departmentName "women ' s hospital".
```

```
  ? x0: departmentLocation? x1.
```

```
}
```

The method used to generate SPARQL is almost the same, the entity and relationship will be extracted from the question sentences and when the question is matched with any template, the query language will be generated and sent to Jena endpoint for searching.

7 IMPLEMENTATION ON NAO ROBOT

In this thesis, the Nao Robot is the platform used to implement this QA system. Nao is a humanoid robot which can interact with user. The core control of NAO is NAOqi. Through NAOqi, NAO is able to do all kinds of actions and actions. At the same time, it is also the tool used to program with different programming languages. Figure 33 shows the control process of the NAO Robot.

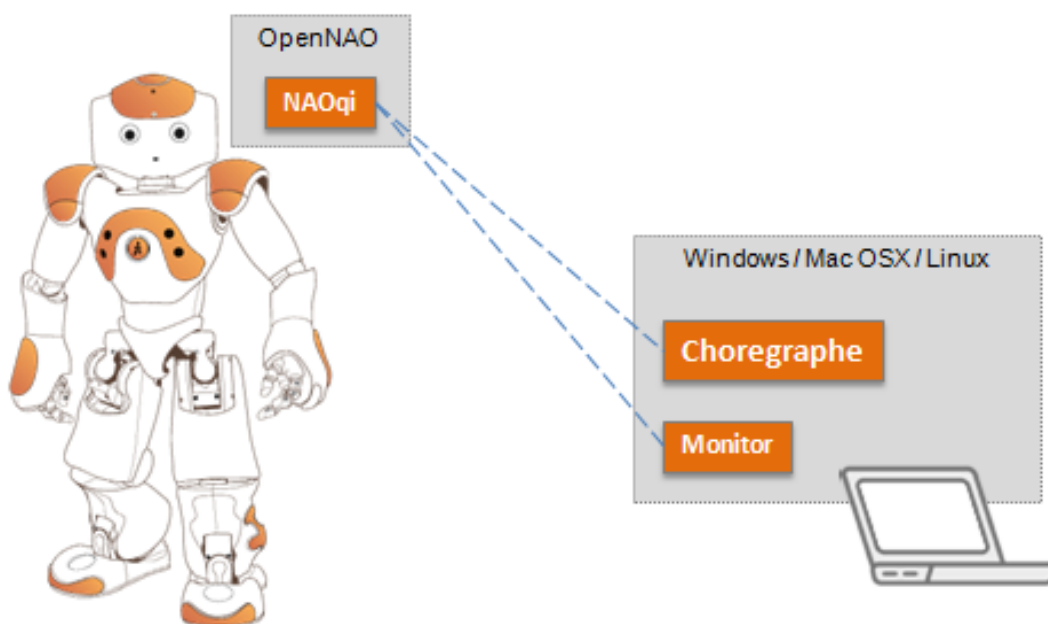


Figure 33. Control Process of NAO Robot/16/

7.1 Speech Recognition

The sound file acts as input of the system. The robot needs to hear what a user says in order to answer questions. The Nao Robot provides “ALAudioRecorder” module from ALProxy to record a wav file when the user is speaking. Figure 34 is the code of how to use this module:


```

global tts, audio, record, aup
robot_IP = ip
robot_PORT = 9559
# -----> Connect to robot <-----
tts = ALProxy("ALTextToSpeech", robot_IP, robot_PORT)
audio = ALProxy("ALAudioDevice", robot_IP, robot_PORT)
record = ALProxy("ALAudioRecorder", robot_IP, robot_PORT)
aup = ALProxy("ALAudioPlayer", robot_IP, robot_PORT)
# -----> recording <-----
print 'start recording...'
record_path = '/home/nao/recording.wav'
record.startMicrophonesRecording(record_path, 'wav', 16000, (0,0,1,0))
time.sleep(i)
record.stopMicrophonesRecording()
print 'record over'

```

Figure 34. Audio Recorder

As the code in Figure 35 shows, the wav file is saved in “/home/nao” path when the sound recorded by the microphone of the NAO robot. By using FTP (File Transfer Protocol), Nao robot is connected with our computer and transfer the wav file to computer. Figure below shows sample code to transfer file between the robot and computer:

```

from ftplib import FTP

def ftpconnect(host, username, password):
    ftp = FTP()
    ftp.connect(host, 21)
    ftp.login(username, password)
    return ftp

def downloadfile(ftp, remotepath, localpath):
    bufsize = 1024
    fp = open(localpath, 'wb')
    ftp.retrbinary('RETR ' + remotepath, fp.write, bufsize)
    ftp.set_debuglevel(0)
    fp.close()

if __name__ == "__main__":
    ftp = ftpconnect("192.168.1.110", "nao", "nimdA")
    downloadfile(ftp, "record.wav", "record.wav")
    ftp.quit()

```

Figure 35. Transfer wav File

After getting the voice file, Google Cloud Speech API is used as an engine to transform voice into text. The ability to convert speech to text is based on deep neural networks, and the latest machine learning algorithm recently proved to be particularly effective for pattern detection in video and audio signals. As Google collects new voice samples, the neural network will be updated to learn new terms and improve recognition accuracy.

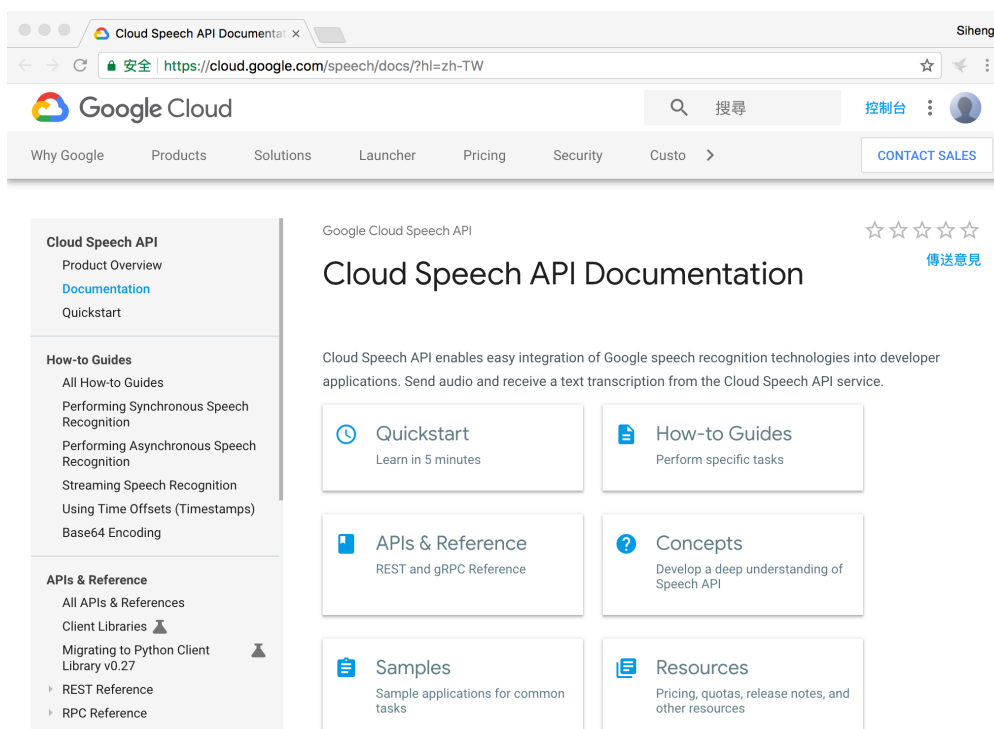


Figure 36. Google Cloud Speech API

In order to use this API, firstly an account of Google cloud platform is needed. After creating a new project in Google cloud platform, Google Cloud Speech API should be enabled in the API Manager. A Speech API request JSON file will be download automatically when a API Key is created and this file should be configured as an environment file in this project. The wav file is sent to the cloud platform and it will be transferred into text which is the return value. The returned text is used for checking whether the robot gets the right question from the user. The code which is shown in Figure 37 shows how the API is used in this project.

```

def transcribe_file(speech_file):
    """Transcribe the given audio file."""
    from google.cloud import speech
    from google.cloud.speech import enums
    from google.cloud.speech import types
    client = speech.SpeechClient()

    with io.open(speech_file, 'rb') as audio_file:
        content = audio_file.read()

    audio = types.RecognitionAudio(content=content)
    config = types.RecognitionConfig(
        encoding=enums.RecognitionConfig.AudioEncoding.LINEAR16,
        sample_rate_hertz=16000,
        language_code='en-US')

    response = client.recognize(config, audio)
    # Each result is for a consecutive portion of the audio. Iterate through
    # them to get the transcripts for the entire audio file.
    for result in response.results:
        # The first alternative is the most likely one for this portion.
        text='{}'.format(result.alternatives[0].transcript)
        print text
    return text

```

Figure 37. Google Cloud Speech API Code

7.2 Result

After recognizing the voice, the text of the question will be sent for processing before it is confirmed by the user. And then, the question will be converted to SPAROL to search from the knowledge base. The Answer will be returned to the Nao Robot. "ALTextToSpeech" module in ALProxy is used to a synthetic speech and answer to the question. Figure 38 is the humanoid NAO robot in laboratory and Figure 39 is part of the print result. When a user asks a question, it will be converted to text. If the text is right, a query will be generated according to this question. After getting the answer, the NAO robot will talk to the user and complete an interaction.



Figure 38. Nao Robot

```

where is Helsinki Hospital
start recording...
record over
yes
PREFIX : <http://www.semanticweb.org/eva66/ontologies/2018/2/helsinki#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?x1 WHERE {
  ?x0 :departmentName "helsinki hospital".
  ?x0 :departmentLocation ?x1.
}

Bulevardi 22 00120 Helsinki

```

Figure 39. Result

8 RECOMMENDATION FOR FUTURE RESEARCH

● IMPROVING WEB CRAWLER TECHNOLOGY

In this project, the web crawler technology is used to gather data from a HTML of the web page. However, for some pages whose structure is not so clear and standard, it will be difficult to process it. Maybe in the future, with the development of this technology, it will be much easier to get the data that we need from the web page. Another problem is when there are many pages need to be crawled, it takes time to process it one by one. The speed of the web crawler can be improved in future research.

● THE SIZE OF KNOWLEDGE BASE

Limited to the author's ability, many works still fail to achieve the desired results. For example, the scale of the knowledge base is not large enough, and the coverage is not wide enough. It just contains information of a few hospitals in Finland and about such things as the location of the department, opening time, telephone number and some data of diseases that we add manually. In future research, the data should be more detailed and authentic, the information about the doctors in a hospital could be added.

● QUESTION TYPE

Currently, there are only seven types of questions can be answered in this system. Each of them has their own question template. For some questions, we cannot give satisfactory results. And the complexity of the problem is not high enough to answer the first and two other problems. In future research, a classification algorithm for machine learning can be used to process text from the website and question directly. Understanding semantic information would be more flexible than just matching the question with a regular expression. Due to the limitation of the author's ability, much of the work fails to achieve the desired results.

9 CONCLUSION

In the open domain (open domain) the automatic question answering system is facing a lot of difficulties, the automatic question answering system in the limited field is indeed being commercialized. Many enterprises, schools, scientific research units, and even parts of websites, such as e-commerce, establish their own question and answer system, which saves a lot of human costs and many common problems can be answered by the machine itself, it is in other words labor-saving.

For the platform of the QA system, there is no doubt that the humanoid robot is the best choice. With the continuous progress of technology, humanoid robots will gradually enter into thousands of families, helping people to deal with many daily things, such as housework, electrical appliances, services, chatting and so on.

In this thesis, the process and history of automatic question answering is introduced, and also the whole process of building automatic question answering system. The process of how to build a question answering system on a humanoid robot is introduced in detail. Crawling of data, information extraction, and the construction of the knowledge base are described. Finally, the question answering system is implemented on the Nao robot and online speech recognition is introduced. At the same time, in order to make NAO more intelligent, it has many human's characteristics, such as "reasoning" and other functions.

REFERENCES

- /1/ Z. Zheng, "AnswerBus Question Answering System," In Proceedings of Second International Conference on Human Language Technology Research, San Francisco, CA, USA 2002.
- /2/ Question Answering
https://en.wikipedia.org/wiki/Question_answering/
- /3/ Python
<https://www.python.org/doc/>
- /4/ Masanès, Julien (February 15, 2007). Web Archiving. Springer. p. 1. ISBN 978-3-54046332-0. Retrieved April 24, 2014.
- /5/ Apache Jena
[https://en.wikipedia.org/wiki/Jena_\(framework\)](https://en.wikipedia.org/wiki/Jena_(framework))
- /6/ "Preface". www.nltk.org. Retrieved 2016-06-15.
- /7/ Yu Tiecheng. The current development of speech recognition [J]. Communication World, 2005.
- /8/ Comparing Speech Recognition Systems (Microsoft API, Google API And CMU Sphinx), March, 2017
- /9/ "NLTK Courses". Google Docs. Retrieved 2016-06-15.
https://docs.google.com/document/d/1eYubSwLkpB7ZgfQVxxAw-gsmAqS__BRfbMyP9qV6ngD8/edit/
- /10/ "Unveiling of NAO Evolution: a stronger robot and a more comprehensive operating system". Aldebaran Robotics. 2014. Retrieved 1 February 2015.
- /11/ "For education & research". SoftBank Robotics. Retrieved 2016-09-30
<https://www.ald.softbankrobotics.com/en/solutions/education-research>.
- /12/ Web Ontology Language
<https://www.w3.org/OWL/>
- /13/ Requests: HTTP for Humans
<http://docs.python-requests.org/en/master/>
- /14/ Gigabotics Robotics and Development and Research

<http://gigabotics.com/robotics/features-of-nao-robot/>

/15/ Nao(robot)

[https://en.wikipedia.org/wiki/Nao_\(robot\)](https://en.wikipedia.org/wiki/Nao_(robot))

/16/ Software in and out of the robot

http://doc.aldebaran.com/1-14/getting_started/software_in_and_out.html

/17/ NAOqi Framework

<http://doc.aldebaran.com/1-14/dev/naoqi/index.html>