JOSAR, Vol. 1 No. 2 September, 2018; p-ISSN: 2502-8251; e-ISSN: 2503-1155

https://ejournal.unisbablitar.ac.id/index.php/josar

SIMPLE CONVERSATION SYSTEM ON SOCIAL ROBOTS WITH LEVENSHTEIN ALGORITHM

Haris Yuana
Computer System Lecturer at FTI Balitar Islamic University
Email :harisyuana@unisbablitar.ac.id

ABSTRACT

Robots in the industrial era 4.0 are not only required to be able to help or replace human work. Humans have positioned robots not only as machines, but also as friends. Like a friend, the robot must be able to interact, communicate, and respond. This phenomenon raises the term social robot, which is a robot that can interact socially like humans do with each other. Robot interaction with humans can be verbal communication (conversation), text, social-media, physical movement, or artificial intelligence (AI). This paper focuses on discussing social robot interactions in (verbal) conversations. Robot conversation patterns can be built simply by applying the Levenshtein algorithm to the database search method to get conversation responses. With this method the robot can provide a response that relates to the topic that the opponent is talking about without going too far.

Keyword ; Social Robot, Conversation System, Interaction, Levenshtein

1. INTRODUCTION

The development of robots is now very advanced. Robots are not only functioned as machine workers that can replace or alleviate human work. Robots have developed into various fields such as education, entertainment, tourism, and households. One of the current trends is the development of robots for households or society. This robot is often known as social robot.

Social robots are robots that interact with humans to follow behavioral norms according to the expectations of their interaction partners. In interacting robots can use sound media, movement, text, or even social media. One important part of supporting communication is voice conversation. This is quite important because most of the patterns of interaction carried out by humans are through sound. The ability of robots to recognize and process sound will be in accordance with the model of human communication and make it easier for humans to interact without having to learn it first.

The use of a conversation system can be applied to various types of robots, for example humanoid robots, animal robots, or on customer

service systems via telephone. One way to implement a robotic conversation system is to apply the Levenshtein algorithm. With this algorithm, the robot can recognize the topic being discussed by the opponent and determine the response that fits the topic.

This paper discusses how to apply the Levenshtein algorithm in a robot conversation system. Basic knowledge of robots is stored in a database. The database contains various topics of normal conversation. When the robot communicates, the robot cannot recognize which topic and what response to take to answer the opponent. The Levenshtein algorithm will help the robot compare the contents of the opponent's conversation with the basic knowledge possessed by the robot. After knowing the weight of the opponent's conversation, the system will try to recognize the topic being discussed and determine the response that best fits the topic of conversation.

The writing of this paper is divided into several sections as follows. Chapter 2 contains a literature review of the robotic conversation system. Chapter 3 describes the design of a conversation system architecture. In chapter 4, the implementation of the robotic conversation system and its configuration are discussed. While chapter 5 contains conclusions and further development plans.

2. LITERATURE STUDY

2.1 SAPI 5.3

SAPI 5.3 is a library made by Microsoft for speech recognition and Text To Speech (TTS) applications. SAPI engine is divided into two types, namely recognition engine and TTS engine. Recognition engines are used to convert human voices to tech. While the TTS system is used to convert text into artificial sounds (sound synthesis).

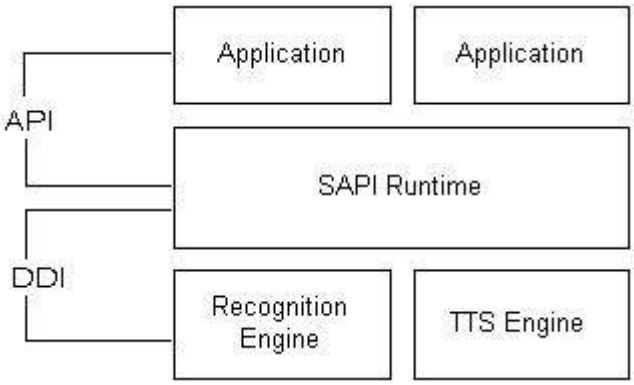

Figure 1. SAPI 5.3 Architecture

2.2 Levenshtein Algorithm

The Levenshtein algorithm is an algorithm to calculate the Levenshtein distance or edit distance from two strings or words. The edit distance value states that many changes are made so that the first string is the same as the second string. Changes made include three types of operations, namely insertion, deletion, and substitution.

Insertion or insertion operations are operations of entering new characters in strings. For example, changing the word "EMPAT" to "TEMPAT" by adding a character "T" to the first word. While deletion or deletion operations are operations by removing one or several characters so that the first string is the same as the second string. For example, changing the word "TEMPAT" to "EMPAT", that is by removing the character "T" at the beginning of the word.

The last operation is substitution or exchange, which is an operation that exchanges the value of a character in a first string so that it is equal to the second string. For example changes to the word "SABAR" become "KABAR". This change can be obtained by exchanging the value of the character "S" to the character "K". In this operation, we can get the edit distance value which is one (1) because only one exchange operation is done.

The edit distance search process can be explained through the following example. For example, look for a distance from the string

"BABY" with the string "BOBBY". The search for distance can be seen in Figure 2. The number value indicates the value of the edit distance.

		B	A	B	Y
	0	1	2	3	4
B	1	0	1	2	3
O	2	1	1	2	3
B	3	2	2	1	2
B	4	3	3	2	2
Y	5	4	4	3	2

Figure 2. Calculate Edit Distance

Each string that is searched for edit distance must be added to an empty character before the first character of the string. If seen in the second row, which is in the line that has the character "B" from the word "BOBBY", it can be seen that to change the n characters in the row to be equal to m characters in the column several steps are needed. For example, changing 5 characters in the word "BABY" so that it becomes 1 character in "B" (from the string "BOBBY") it takes 3 operations. The operation is three times the deletion in the combination of the characters "ABY". Similarly, the next distance calculation step, until finally the final result is obtained that to change the character "BABY" to "BOBBY" 2 steps of operation are needed. That is one substitution operation by changing the character "A" to "O", then one insertion operation by adding the letter "B". This value is then the edit distance value from the string "BABY" with the string "BOBBY".

3. DESIGN

3.1 Hardware

In hardware architecture there are two main components, namely robots and servers. The conversation application will run on a server connected to the robot. For voice, the server will process voice input from robots sent wirelessly. These inputs are processed on the server to get the appropriate response. Various forms of response are stored in the database as basic knowledge of robots. The server will also be connected to the internet to get the latest weather and news information.

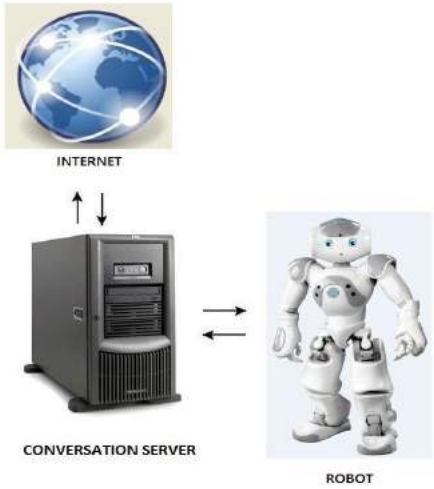

Figure 3. Hardware Architecture

3.2 Software

The software architecture is only a block of the Conversation System. The Conversation System is used to process voice input. The incoming sound input will be converted to text or string. Then the program will use the Levenshtein algorithm to calculate Levenshtein distance between input and database. From the comparison process, the smallest results are taken so that the robot can find the most appropriate response. The response that has been obtained is then

processed into the TTS Engine from SAPI 5.3 to produce a response in the form of sound.

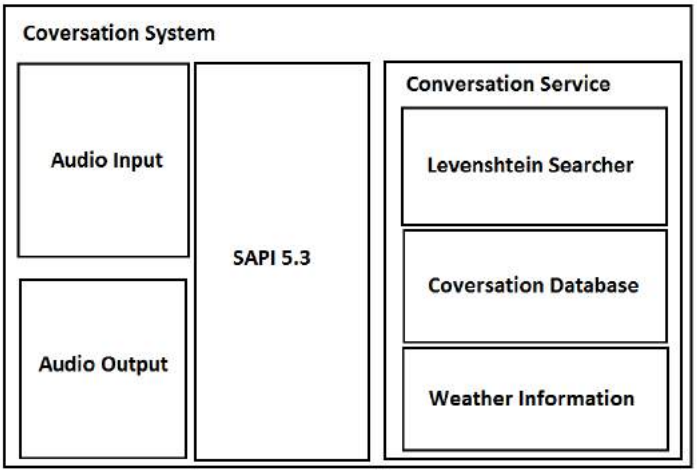

Figure 4 Software Architecture

3.3 Database

The main database conversation system consists of only two tables with one-to-many structures. The table is Key Word Tbl and Response Tbl. Key Word Tbl contains a variety of topics that become benchmark references to inputs. The Levenshtein algorithm will measure input value for this key table. The smallest Levenshtein distance value will be selected in response to the robot. Response in one key can be more than one. This allows the robot to give a varied response even though talking about the same topic.

Figure 5. Database

4. Implementation

The conversation system is implemented in the dot Net (.Net) framework using the C # programming language. Program development is done using the Integrated Development Environment (IDE) of Visual Studio 2010. For ease of organization, the program is divided into six solutions, namely Conversation System and WCF Language Processing. Applications that have been completed and successfully compiled are then deployed on the server. For applications in the form of websites and services, they are deployed through the Internet Information Service (IIS) Manager, which is an application server for hosting websites and services on Windows 7 or Windows 8.1 operating systems. The process of deploying and configuring Task Scheduler can be seen in Figure 6.

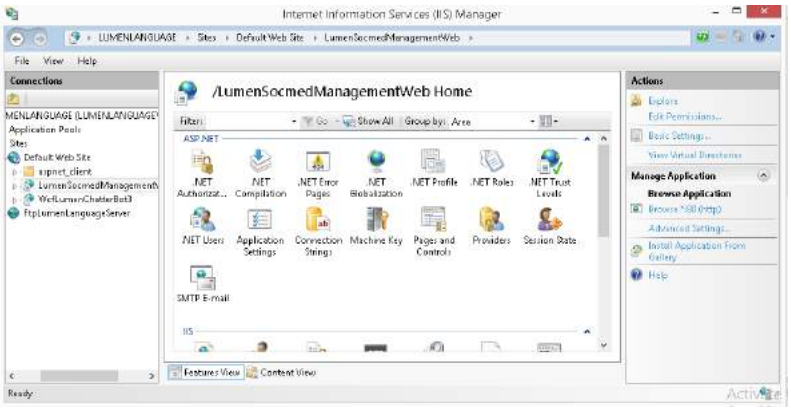

Figure 6. IIS Manager

Conversation System solution functions to process input and output in the form of sound using Microsoft SAPI 5.3. This system will access the service from the WCF Language Processing program. The vote that has been converted to text is then sent to the service to get a response. After obtaining the appropriate response based on the Levenshtein algorithm, the next response is sent to the TTS engine in SAPI 5.3. The next robot speaks according to the response text obtained.

5. Conclusion and Future Work

In this paper the language of how to build a robotic conversation system is simple by using service, database, and speech recognition and TTS libraries. In experiments conducted robots can respond satisfactorily and be able to adjust the topic. The response depends on the process of converting the input sound into text. When there is a lot of noise, robots often make mistakes in the conversion.

The ability of robots to recognize topics and look for responses depends on the ability of library speech recognition to be used. In addition, the system does not yet have conversational intelligence similar to humans. This happens because the response given is reactive. The system has not been able to save the state of conversation that has been done before.

Further development of the conversation system can be focused on the addition of artificial intelligence that is capable of storing the states of conversation. With this model the system is likely to be able to respond to the conversation well and more actively. In addition, the development of programs through multithreading will improve the system response better.

REFERENCES

[1] Agafonov, Eugene, Multithreading in C# 5.0 Cookbook, Packt Publishing, Birmingham-Mumbai, 2013.

[2] Caporael, L. R., Three Tips from a Social Psychologist for Building a Social Robot, 2006.

http://ieeexplore.ieee.org/ielx5/10858/34213/01631753.pdf?tp=&arnumber=1631753&isnumber=34213, 7 Maret 2014, 10.31 WIB.

[3] Hegel, F., Muhl, C., Wrede, B., Hielscher-Fastabend, M., & Sagerer, G., Understanding Social Robots, 2009. http://ieeexplore.ieee.org/ielx5/4782474/4782475/04782510.pdf?tp=&arnumber=4782510&isnumber=4782475, 7 Maret 2014, 10.37 WIB.

[4] McMillan, Michael, Data Structure and Algorithms Using C#, Cambridge, New York, 2007.

[5] Vernica, Rares, and Li, Chen, Efficient Top-k Algorithms for Fuzzy Search in String Collections, http://www.ics.uci.edu/~rares/pub/keys09-p9-vernica.pdf, 10 Mei 2014, 10.41 WIB.

[6] _____, Speech API Overview (SAPI 5.3), 2014. http://msdn.microsoft.com/en-us/library/ms720151%28v=vs.85%29.aspx, 19 Mei 2014, 11.30 WIB.

[7] _____, Microsoft Visual Studio 2010 Service Pack 1 (Installer), 2014. http://www.microsoft.com/en-us/download/details.aspx?id=23691, 11 Juni 2014, 10.43 WIB.

[8] _____, Hardware and Software Requirements for Installing SQL Server, 2014. http://msdn.microsoft.com/en-us/library/ms143506.aspx#hwswr, 11 Juni 2014, 10.55 WIB.

[9] _____, Vladimir I. Levenshtein Biography, 2014. http://www.ieeeghn.org/wiki/index.php/Vladimir_I._Levenshtein, 19 Mei 2014, 11.30 WIB.