A Framework for Multilingual Real-time Spoken Dialogue Agents

Arnaud Jordan and Kenji Araki
Graduate School of Information Science and Technology
Hokkaido University
Sapporo, Japan
{arnaud, araki}@media.eng.hokudai.ac.jp

Abstract—**In this paper, we propose a framework for a spoken dialogue agent that is not dependent on any specific language; it takes some dialogues and sentences as training sets and uses them to acquire knowledge about the target language, then it uses this knowledge to generate several possible responses corresponding to the user input and finally it uses a simple score method to select the best one to show to the user. In aim to be language independent the system only uses very basics treatments and combines them to generate the output sentences. Moreover, all the learning and generation processes are realized in independent threads making the system enable to generate the outputs in real-time. Concretely, the user can input a new sentence at any time and influence the current output generation. We carry out experimentation in two grammaticality different languages and got some results proving our system is efficient to generate responses of a simple dialogue.**

Keywords—spoken dialogue agent, multilingual, multi-thread, real-time, machine learning, natural language processing

I. Introduction

Nowadays, many spoken dialogue agents have been proposed. Most of them focus on one language and use language specific resources such as grammar rules, word relation database like WordNet[1] or morphological analysis tools such as Juman[2] for Japanese. Many algorithms using those kinds of resources have been proposed and some of them can handle very pleasant dialogue such as ALICE[3]. We also developed some agents[4][5], which are only available for Japanese. Nevertheless, the most frequently used resources are available in different languages. In consequence, the same system can be adapted to different languages using specific resources for each of them[6]. In this case the system is multilingual, but cannot be adapted easily to language for which the used resources are not available.

However, humans are able to learn and to speak any language even if they do not know some basic grammar rules or word functions such as articles especially in the case of their native language. They can also choose to transgress some rules to make sentence more natural. They just need to be immersed in a specific language to learn it. That is why; we consider that an algorithm can generate correct responses without any use of language specific grammar concepts, but only using statistical relations. Concretely, to create a more human like system we developed a framework for multilingual spoken dialogue agent, which can generate outputs in any language. As language resources, the framework only uses dialogue samples and sentence corpus without any language related annotations. In addition, the framework implements a highly threaded approach to make it real-time. Each treatment is realized in a specific thread and in parallel to other treatments.

In the second section of this paper, we explain the framework's outline. Then, we present the settings and the results of the experimentation we carried out. Finally, we conclude about drawbacks and future development of the presented framework.

II. Outline

One of the main aims of the framework presented is to be language independent. In consequence, it uses no language specific grammar rule sets, dictionaries or morphological analysis tools, but it directly acquires the needed knowledge from two resources, dialogue samples and sentence corpus.

This framework is called MRDF in the next sections.

A. Data management

In aim to develop a real-time system, the user input and the data generated from it have a lifespan and finished by disappeared. Concretely, each element have 50% chances to be removed after a definite time, which is called the lifespan time.

In addition, to be able to evaluate and to select useful element, each of them has a score value. This value decreases with the time and tend to fluctuate between 0 and 1. This fluctuation is a kind of Zero-point energy used to make system livier. If the score is under 1, the element cannot be accessed.

Fig. 1 shows the implementation in Java[1] of the data lifespan.

```
public void run() {
    if(this.getNote()>0) this.score/=2;
    else this.score++;
    if(rand.nextFloat() < keepInLife) return;
    this.died();
}
```

Fig. 1. Implementation of the data lifespan's process

[1] http://java.sun.com

In addition, the system also saves the ancestors of all the elements for some optimizations and simplifications purpose (cf. II.B.2). The ancestors are the elements from which the element has been generated.

1) Data access

In the framework, data are not sorted and never compared each other; the only way to access an element is to ask the system to provide a randomly selected one. However, an element with a higher score is selected more often than a one with a lower score, a score of 0 makes the element not accessible.

B. Basic treatments

We identify 5 basics treatments to generate a response. All the outputs of MRDF are generated while combining them. We list them below.

1) Splitting

It consists in generating sub-element of an initial element. This treatment is similar to a morphological analysis. Nevertheless, the system does not know any information about the language it treats; it cannot identify the different words of the sentence. In consequence, it has to split an element using the knowledge it acquired previously. Concretely, it uses the dialogue samples as initial cutting points. For example, if we teach the system the two following sentences "I like cookies" and "I like", the system can use the second to split the first and get the two substrings "I like " and "cookies". Then, if the system does this process continuously and uses the newly generated substrings as cutting points it can learn many substrings, which can be used to split the user input.

The generated sub-elements' score is equal to the initial element, but their lifespan is half of it.

In addition, when an element is split into two or three sub-elements, the system acquires a relation between them. For example, when the string "Now I prefer dog." is split using the string "I prefer", the system acquires the two relations {"Now", "I prefer"} and {"I prefer", "dog"}. Those relations are equivalent to a bigram[7]. Moreover, the system also acquires two substitution rules (cf. II.B.3).

2) Merging

It is the opposite of the splitting process, merging consist in merging two elements into one. To avoid meaningless merging, the system only merges two pairs, which was results of a splitting (which are related) such as "I like " and "cookies".

Merged elements' score and lifespan are the means of the two proceeded elements.

a) Ancestor

Splitting and merging are two opposite treatments, merging two elements generated from the same initial element is not useful and create duplicate element, to avoid this kind of process, the system checks if the merging result is one of the ancestors of one of the two elements merged, if it is the case, they are not merged.

3) Substitution

Substitution changes one element into another without removing the first one. The aim of the substitution is to represent the connection between different conceptual entities. We think it is a similar process to the association[8] in psychology. All concepts are related through substitutions. For example, "Strawberry" can be related to "are tasty". In consequence, if the dialogue contains the string "Strawberry", the system will generate the string "are tasty". If other treatments tend to generate "are tasty", the system will have more chance to access this element, merge it with "Strawberry" and finally select the sentence "Strawberry are tasty" as the output.

The score and the lifespan of the substituted element are the half of the original one.

Besides, in a similar way as graph clustering[9], this relation can be used to discover words clusters. For example, the word "animal" will tend to have many substitutions to animal words like "cat" or "dog" and other related words like "elevate". In consequence, if the question is "What animal do you like?", the interrogative pronoun "what" will be substituted to many different words, but "animal" and "like" substitutions will create two sets of words and we suppose the intersection of this two sets will be a good answer such as "cat" or "dog".

a) Pack

Some substrings such as "?" are very frequent and generate many substitutions[2], which are often not valuable for the output generation. To solve this problem, we implement a similar method as the tf–idf method[10]. The system ponders the value of each substitution, more an element is rarely substituted more it is valuable. Concretely, all substitutions of the same element are put in the same pack, which has the same access rate as a unique element.

4) Voting

To select and identify the most valuable element, the system attaches them a score, voting consists in increase the score of an element, which matches some criterions such as an element, which matches a previously acquired knowledge.

5) Selecting

Selecting is the most important phase of the output generation; it selects which element to output and show to the user. If an element is selected, it will be removed from the system to avoid multiple selections of the same element.

Moreover, to avoid incorrect sentence's outputs, we do not allow any output, which is not contained in the sentence corpus. However, in the case of a complete implementation of a spoken dialogue agent the system have to allow some outputs which are not present in a corpus, but which can be considered as correct. For example, if there are two samples "Cats are animals" and "Animals are living beings", the system must be able to generate the output "Cats are living beings" even if the sentence is not present in the sentence corpus. Nevertheless, for this paper experimentation we limit to corpus exact match.

[1] The question mark is related to all question of the corpus.

a) Trigger

The selecting process asks the system for a random element and if its score exceeds a trigger value, the system will output it. In aim to create a real-time system, like a human the system has to reply in a minimum of time, but with a maximum of pertinence. To produce this kind of behavior the system uses a dynamic trigger, which value decrease in function of the time spent. Fig. 2 shows an example of the trigger evolution.

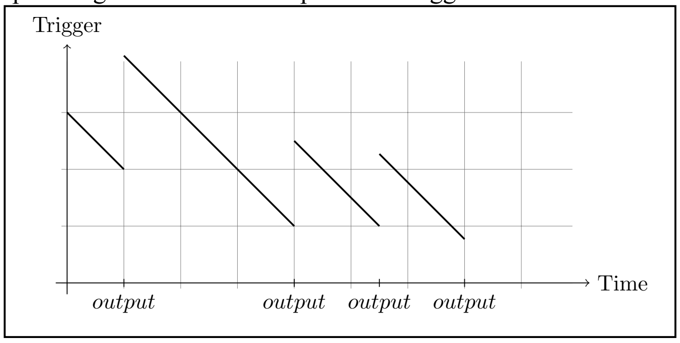


Fig. 2. Example of the trigger evolution

After the system selected an output, the trigger value is reinitialized. This value is calculated using (1). Concretely, the system uses the mean of the last five outputs trigger value to calculate the new trigger value. This method let the system adapts to the elements' score automatically.

$$V_i = (\sum_{k=i-5}^{i-1} V_k)/5 * 2 \qquad (1)$$

- V: value of the trigger
- i: output number

C. Parallel processing

All basics treatments are performed in a specific thread. In consequence, the system is composed of more than thousand of threads; to deals with this lot of processes we used the Java Executor API[3] thread pool to limit the number of effective threads.

The agents can access one or several existing data and also create new data. For example, an agent can access the two data "Today is " and "Monday" to generate the new data "Today is Monday". This process will not block any other process that will be executed at same time.

No treatment is considered more important than another one; we let the system choose the best combination of rules by combining them randomly. We consider this behavior like a kind of intuitive and experiential processing. The merit of this behavior is to have a high fault tolerance, even if an error occurs the output generation will continue proceeded by other agents. Moreover, adding and removing a process do not require modifying existing processes.

1) Sleeping time

Acquiring splitting and substitution rules are time and computation consuming. That is why the system stops this process when it receives an input and allows all the capacities to the output generation. If there is not any input to proceed, the system will enter in sleeping time and start treatments to try to acquire new knowledge from previously acquired sentences. This learning process is continuously proceeded.

D. Training samples

Our system uses no language specific processes or resources. Nevertheless, it uses two kinds of training set, dialogue samples and knowledge samples. These resources only contain natural language sentences and some tags about the environment, but no information about the language itself.

The number of agent increase in function of the size of the training samples

1) Dialogue samples

The dialogue samples contain some very simple dialogues used to acquire substitution and splitting rules in the target language. Moreover, the dialogue samples contain some tags about the conversation's environment. They are used to make the conversation livelier and to facilitate the real-time output generation. We list those tags in the next sections. Fig. 4 shows an example of dialogue samples containing some tags.

```
<Silence>
U1: Let's speak about your hobbies.
U2: I like reading.
<Context clear>
U1: Is it good?
U2: It is good.
```

Fig. 3. Example of dialogue sample

a) User

The *User* tag[4] indicates which user said the sentence; some sentences are responses to another user and others follow the same user previous sentence.

b) User entrance

To make the system livelier, we want it to be able to display a sentence after the user enters in the chat room, but before he inputs anything. To implement this behavior, the system generates a *User entrance* tag when the dialogue starts.

c) Context clear

Dialogue samples do not contain only one dialogue example, but several dialogues. In consequence, the system has to be informed when the dialogue changes to not create any link with the previous dialogue, which is over and has no relation with the newly starting dialogue. The *Context clear* tag is used to indicate this situation.

[2] http://docs.oracle.com/javase/tutorial/essential/concurrency/exinter.html

[3] "U1" and "U2" in Fig. 4.

d) Silence

Silences are an important part of a dialogue and in consequence have to be indicated to the system when they happen. This information can be used to generate an output having for aim to continue the dialogue with the user such as "Let's speak about your hobbies." when a silence occurs.

e) Reflection

When many substitutions occurred in the system, we consider it in reflection phase and a *Reflection* tag is generated. This tag can be substituted to some interjections indicating the reflection's process. For example, "err" can be outputted to indicate that the system is thinking and need more time before answering.

f) Void

In opposition to the *Reflection* tag, the *Void* tag indicates the system is not thinking and is not generating any sentence. This information can be used to indicate the user to not wait any supplementary output.

2) Knowledge acquisition

The dialogue samples contain a sentence and one corresponding correct response. Using these two sentences the system can acquire more substitution rules. For example, the sentence "Do you like coffee?" and the response "I like coffee" can be used to generate the substitutions {"Do you like", "I like", "?", "I like"}, which are essential for question answering.

3) Knowledge samples

The knowledge sample is a simple sentence corpus without any tag. It contains some basic knowledge and it is used to acquire some substitution rules and for selecting the output.

III. Example of the output generation

As explained above, for this research we aim to generate sentences, which can match a sentence present in a sentence corpus. The output generation processes are all executed at the same time. However, Fig. 5 shows the output generation processes main phases in a logical order.

Input: Do you like drinking milk?

Splitting: {"Do you like", "drinking milk?", "drinking milk", "?", ...}

Substitution: {"I like", "I do not like", ...}

Merging: {"I like drinking milk", "I do not like drinking", ...}

Voting: {"I like drinking milk"}

Selecting (output): I like drinking milk

Fig. 4. Example of the output generation process

The generation process does not stop after outputting the first response, but continues while the system contains data generated from the user input. That is why the system can output several responses to one single question. In addition, during this process the user can input other sentences, which will be aggregated to the currently processed data and influence the output results.

IV. Language difference

In language like English or French, the subject of the sentence is explicitly indicated for example using "you" or "I". However, in Japanese this subject is generally implicit. In consequence, to generate a response to a question in Japanese, often only one splitting removing the question marker part is needed, but in French one splitting, one substitution and one merging is needed. Fig. 6 is an example of this situation. In Japanese, the output is equal to the input removed from " ka?".

Tu aimes le chocolat ?	→	J'aime le chocolat
Choko ga sukidesu ka?	→	Choko ga sukidesu
(Do you like chocolate?	→	I like chocolate)

Fig. 5. Example of a simple question in the two target languages

V. Experimentation

The system developed is only a base framework and does not have for objective to be better than a system, which focus only on one or several languages. The objective of the experimentation we carried out is to prove that the same system can generate natural responses efficiently in several languages using the same basic treatments. Concretely, the system uses the rules it acquired from dialogues and knowledge samples to generate an output.

We think any spoken dialogue agent has to objective to generate an output, which matches some patterns to be considered as correct. More the system is developed more these patterns become complicated and can contain meta-model or sets of possibilities. That is why, for our framework first evaluation we choose to use the simplest pattern, which is complete correct sentence.

As baselines we use a similarity-based system in a similar way as Murata's system[11]. However, the proposed system uses some Japanese grammar particularities to calculate the similarity between two strings. In our case, we had to use generic string similarity algorithm to be applicable for any language. Concretely, we use Jaro-Winkler[12] and Levenshtein[13] distance algorithm. We consider them as the most frequently used distance. The baseline systems select as output the sentence present in the knowledge corpus, which is the closest to the input. We use the Java SimMetrics[5] implementation of these two algorithms.

In the next part of the paper, the baseline system using the Jaro-Winkler distance will be called J-W baseline and the system using the Levenshtein distance is called L baseline.

A. Preparation of the experimentation

First, we prepare sets of inputs and outputs in the two target languages, Japanese and French. Those sets can have a big influence on the experiment result; that is why we selected a very simple and common sentences set for. Moreover, MRDF is not able to treat more complex input yet. The Japanese sentences are directly extracted from our pervious research[13]

[4] http://sourceforge.net/projects/simmetrics/

experimentation dialogues. In this research, the subjects had to ask some questions and if the system cannot answer it, they had to teach the correct answer. We select the questions and the answers taught by the subjects to create the experimentation input set and the knowledge samples. However, we avoid “Yes” and “No” responses, which are too vague and could be used to answer most of the question of the corpus. The French sentences are manually translated from the Japanese one. Both sets contain the same knowledge, but are not literal translation to make the system outputs as natural as possible.

The dialogue samples used for the experimentation are constituted of about 20 simple manually created dialogues.

B. *Experimentation settings*

Using our system and the baseline we generated responses for 100 inputs. Table I shows some information about the inputs and dialogues used for the experimentation.

TABLE I. EXPERIMENTATION SETTINGS

	Japanese	**French**
Question and response	23	21
Input number	100	100
Question number	92	91
Output number	82	84
Number of word	552	526

To determine if an output is correct we ask subjects to evaluate the generated responses. The subjects were native speakers of the system target languages. The Table II shows some information about the subjects. A response is considered as correct if more than 3 subjects evaluate it as correct.

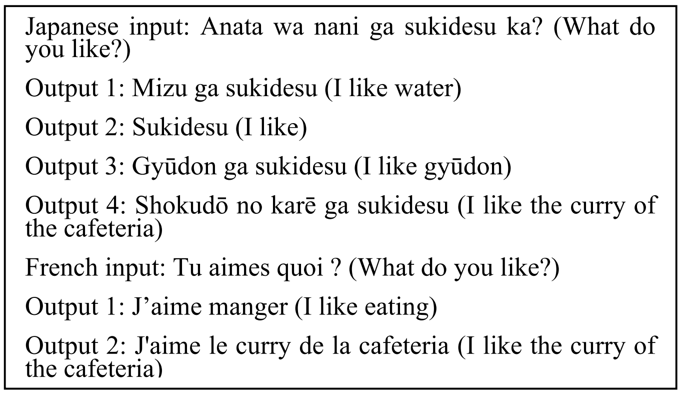
In addition, we also evaluate the number of consecutive correct responses of each system. Concretely, the subjects note the numbers of responses they think are correct starting with the first response.

TABLE II. SUBJECT’S INFORMATION

	Number	
Language	***Japanese***	***French***
Subject	5	5
Male	3	2
Female	2	3
Student	1	3
Worker	4	2
Average age [year]	26.6	22.2

C. *Results*

The three figures below, Fig. 5, Fig. 6 and Fig7 show examples of the generated outputs of MRDF and the two baselines in Japanese and in French. MRDF generated a variable number of outputs in function of the input and the knowledge it acquired previously. Of course, the system can be set to output more or less outputs, but the ratio of correct responses decreases more outputs are numerous. For the two baselines we only wrote the first 3 outputs.


Japanese input: Anata wa nani ga sukidesu ka? (What do you like?)

Output 1: Mizu ga sukidesu (I like water)

Output 2: Sukidesu (I like)

Output 3: Gyūdon ga sukidesu (I like gyūdon)

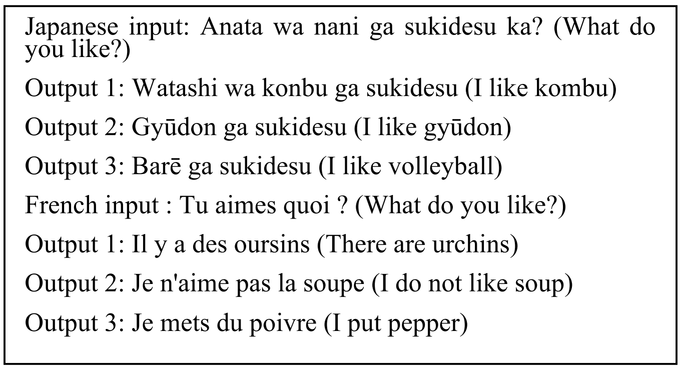
Output 4: Shokudō no karē ga sukidesu (I like the curry of the cafeteria)

French input: Tu aimes quoi ? (What do you like?)

Output 1: J’aime manger (I like eating)

Output 2: J'aime le curry de la cafeteria (I like the curry of the cafeteria)

Fig. 6. Example of responses generated by MRDF


Japanese input: Anata wa nani ga sukidesu ka? (What do you like?)

Output 1: Watashi wa konbu ga sukidesu (I like kombu)

Output 2: Gyūdon ga sukidesu (I like gyūdon)

Output 3: Barē ga sukidesu (I like volleyball)

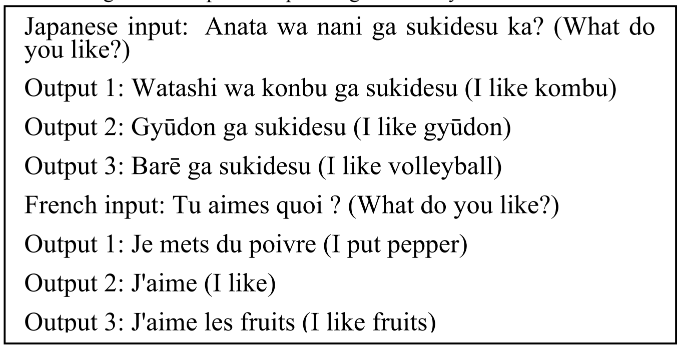
French input : Tu aimes quoi ? (What do you like?)

Output 1: Il y a des oursins (There are urchins)

Output 2: Je n'aime pas la soupe (I do not like soup)

Output 3: Je mets du poivre (I put pepper)

Fig. 7. Example of responses generated by the J-W baseline


Japanese input: Anata wa nani ga sukidesu ka? (What do you like?)

Output 1: Watashi wa konbu ga sukidesu (I like kombu)

Output 2: Gyūdon ga sukidesu (I like gyūdon)

Output 3: Barē ga sukidesu (I like volleyball)

French input: Tu aimes quoi ? (What do you like?)

Output 1: Je mets du poivre (I put pepper)

Output 2: J'aime (I like)

Output 3: J'aime les fruits (I like fruits)

Fig. 8. Example of responses generated by the L baseline

On the previous examples, Fig. 8 and 9, we can see that in Japanese the two baselines select the same responses. However, in French the outputs are a little different. We think it is because of the different sentence structure and writing system between the two languages.

The Table III contains the results of the experimentation for MRDF and the baselines in the two target languages. To calculate the number of consecutive correct responses we only count consecutive responses starting with the first one, which was evaluated as correct by more than 3 subjects. For example, for the French experimentation of MRDF, we got 70 correct first responses, 12 correct second responses and 2 correct third

TABLE III. EXPERIMENTATION RESULTS

Language	Japanese			French		
System	***J-W baseline***	***L baseline***	***MRDF***	***J-W baseline***	***L baseline***	***MRDF***
Total number of responses	8200	8200	172	8400	8400	181
Rate of first correct responses [%]	55.0	58.0	72.0	48.0	56.0	70.0
Rate of consecutive correct responses [%]	64.0	77.0	83.0	69.0	71.0	84.0

responses for a total of 84 correct responses for 100 questions.

D. *Consideration*

Firstly, we can see that the two baselines only order the knowledge corpus in function of the input. In consequence, for each input the entire possible sentences are outputted. This problem can be solved by set a minimal similarity value. MRDF selects only sentence which has a minimal score, in consequence the responses are less numerous.

We can see that MRDF generates more correct responses (71.0%) than the two baselines (51.5% and 57.0%). Moreover, we can also see that MRDF outputs a little more correct response with a number of consecutive correct responses of 83.0% for Japanese and 84.0% for French. Some inputs get no responses; in this case the system can be set up to output some apologies such as “Sorry, I can not answer”.

However, there are still many incorrect responses, especially for the inputs containing interrogative pronouns. If we ask the system “What do you like?” it may output “I like water” as well as just “I like” without any object.

1) *Time generation*

We also measure the time take by the outputs generation of MRDF. For the Japanese experimentation, the mean generation time of the first response was 5.76s and 10.04s for the French experimentation. This time difference can be explained by some simplifications introduce by Japanese implicit subject (cf. IV). However, the mean time generation was 25.00s for Japanese and 17.04s for French.

VI. CONCLUSION

In this paper, we proposed a new language independent real-time framework for spoken dialogue agent that we called MRDF. We also proved that this framework could be used to generate and select correct responses to a simple input in totally different languages (grammatically different and using different script). In addition, the system needs only a simple small starting data set containing basic knowledge to be operational.

However, there are still many works to do, especially to handle interrogative pronouns more efficiently and to be able to answer more complicated questions. For example, the pronoun “who” has to be substituted by any element of the cluster constituted by all persons name or title.

Moreover, MRDF is not ready for scale increase, the number of parallel processes is too important even with a small amount of knowledge. That is why, we are thinking about implementing the substitution part of the framework by using graph instead of a simple agent for each of them.

Finally, in future research, we are thinking about make the framework able to handle more knowledge and to carry out experimentation in additional language to compare the results. Moreover, we will try to add some sentimental treatments to make the framework outputs more consistent. For example if the system hates “peach” we can avoid the generation of the sentence “I like peach” because “I like” is positive and in consequence incompatible with “peach” which is associated to a negative sentiment.

REFERENCES

[1] George A. Miller, “WordNet: A Lexical Database for English”, Communications of the ACM, vol. 38, no. 11, pp. 39-41, 1995.

[2] Kurohashi and Kawahara lab., “Japanese morphological analysis system JUMAN version 7”, Graduate School of Informatics, Kyoto University, 2012

[3] R. S. Wallace, “The anatomy of A.L.I.C.E.”, Parsing the Turing Test, pp. 181-210, 2009.

[4] Arnaud Jordan and Kenji Araki, “Spoken dialog processing for acquiring taste and knowledge”, Proceedings of PACLING2013, 2013.

[5] Arnaud Jordan and Kenji Araki. Comparison of two Knowledge Treatments for Questions Answering. Proceedings of SNLP2013, 2013

[6] Kenji Araki and Michitomo Kuroda, “Generality of spoken dialogue system using SeGa-IL for different languages”, Proceeding of the IASTED International Conference COMPUTATIONAL INTELLIGENCE, pp. 70–75, 2006.

[7] Christopher D. Manning and Hinrich Schütze, “*Foundations of Statistical Natural Language Processing”*, MIT Press, 1999

[8] Wikipedia contributors. "Association (psychology)." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 9 Apr. 2014.

[9] Beate Dorow and Dominic Widdows, “Discovering corpus-specific word senses”, Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2, pages 79–82. Association for Computational Linguistics, 2003

[10] Salton, Gerard, and Michael J. McGill, "Introduction to modern information retrieval.", 1986.

[11] Masaki Murata, Masao Utiyama and Hitoshi Isahara, “Question Answering System Using Similarity-Guided Reasoning”, IPSJ SIG Notes, pp. 181-188, 27 january 2000.

[12] Jaro, M. A., “Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida”, Journal of the American Statistical Association, 84:414–420, 1989.

[13] Levenshtein, V. I., “Binary codes capable of correcting deletions, insertions, and reversals”, Cybernetics and Control Theory, 10(8):707–710, 1966